Titre: Title:	Le coroutage : une approche de routage hybride
Auteur: Author:	Fadi Bizri
Date:	2006
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Bizri, F. (2006). Le coroutage : une approche de routage hybride [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7802/

UNIVERSITÉ DE MONTRÉAL

LE COROUTAGE: UNE APPROCHE DE ROUTAGE HYBRIDE

FADI BIZRI
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(TÉLÉCOMMUNICATIONS)
NOVEMBRE 2006

Your file Votre référence
ISBN: 978-0-494-25530-8
Our file Notre référence
ISBN: 978-0-494-25530-8

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

LE COROUTAGE: UNE APPROCHE DE ROUTAGE HYBRIDE

présenté par: BIZRI Fadi

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. FRIGON Jean-François, Ph.D., président

Mme SANSÒ Brunilde, Ph.D., membre et directeur de recherche

M. GIRARD André, Ph.D., membre

à la mémoire de Sania Tawfic Haroun Bizri

REMERCIEMENTS

Je tiens tout d'abord à remercier ma directrice de recherche, le professeur Brunilde Sansò, pour m'avoir accepté, conseillé et guidé dans le cadre de son laboratoire de recherche. En particulier, je lui suis très reconnaissant pour sa patience, sa confiance et sa générosité durant la phase d'élaboration de mon projet, ainsi que durant les situations personnelles difficiles que j'ai vécues.

Je tiens aussi à remercier particulièrement Christian Awad. Durant ces années passées au GERAD, Christian était à la fois un collègue de travail et un grand ami. Son aide et ses conseils ont été essentiels durant ma recherche, en particulier ses connaissances techniques et son savoir-faire informatique irremplaçables. Les longues heures passées ensemble, que ce soit à travailler, à prétendre travailler, ou durant nos loisirs, étaient toujours imprégnées de bonne humeur et de camaraderie. Je lui souhaite le meilleur succès pour la conclusion de son doctorat.

J'aimerais également remercier MM. Jean-François Frigon et André Girard, pour avoir accepté d'être membres du jury et pour avoir pris le temps de lire, commenter et critiquer mon mémoire.

J'ai aussi très apprécié être membre du GERAD pour la qualité des installations, ainsi que pour le professionnalisme et la gentillesse des membres. Merci en particulier à Pierre Girard pour son soutien technique efficace et aux secrétaires pour leur dévouement et leur bonne humeur. Les secrétaires du département de génie électrique ne sont pas en reste, elles m'ont toujours offert une aide et des conseils rapides, accompagnés d'un sourire.

Merci à tous mes collègues de travail, en particulier Jules, Anne, François, Hatim, Stefano, Cristina et Armelle, pour avoir créé une atmosphère sympathique et agréable au local 4414.

Finalement, un remerciement spécial à mon père et à mes frères, pour leur amour et leur soutien. À ma mère, envers laquelle il n'y a pas de mots pour exprimer

mes sentiments, je ferais de mon mieux pour honorer sa mémoire, et être le fils et l'homme dignes de sa personne et de ses valeurs.

RÉSUMÉ

Le Coroutage est une approche hybride de routage, qui combine les mécanismes des protocoles dits de type État de Lien, ainsi que ceux des protocoles de routage explicite. L'idée consiste à router les flots TCP avec le protocole des états de liens, et les flots UDP temps-réel avec le protocole de routage explicite.

Le protocole état de lien trouve le chemin à coût minimal (CCM) entre chaque origine et destination (O/D) du réseau, en utilisant les coûts des liens comme métrique. Du fait d'avoir un unique chemin entre chaque O/D, il s'ensuit un déséquilibre dans la distribution du trafic des données, avec quelques liens transportant la majorité des paquets.

Le Coroutage cherche à améliorer la performance des flots UDP temps-réel (voix et vidéo) en utilisant le déséquilibre dans la distribution du trafic à son avantage. Ainsi, la partie routage explicite du Coroutage trouve des chemins composés de liens sous-utilisés (en terme de leur bande passante) et envoie les flots UDP sur ces chemins. Les flots TCP, quant à eux, restent sur les chemins à coûts minimaux.

Les résultats de simulation, obtenus avec le logiciel NS2, montrent que le Coroutage est capable d'améliorer sensiblement les délais et la gigue (la variation du délai) pour les flots temps-réel, et ceci sans affecter les flots TCP de façon notable.

ABSTRACT

We propose a hybrid routing scheme that combines traditional Link State mechanisms as well as a new form of explicit routing executed at the flow level. This scheme consists of finding explicit routes, for real-time UDP flows, on network links that are not being used at more than 50% bandwidth utilisation at flow entry time. As for TCP packets, they will be routed on the unique least-cost path found through standard Link State algorithms. We believe our approach can improve the overall robustness and scalability of a backbone network, thanks to its inherent traffic engineering mechanisms, as well as offer better overall service for jitter and delay-sensitive real-time traffic.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES NOTATIONS ET DES SYMBOLES

CCM:	Chemin à Coût Minimal
LS:	Link State
EL:	État de Lien
LSA:	Link State Advertisement
$OSPF$:	Open Shortest Path First
$IS-IS$:	Intermediate System to Intermediate System
TCP:	Transport Control Protocol
UDP:	User Datagram Protocool
O/D:	Origine/Destination
UL:	Utilisation de Lien
QdS:	Qualité de Service
SIS:	Séquence d'Interfaces de Sortie

LISTE DES TABLEAUX

LISTE DES ANNEXES

INTRODUCTION

Les protocoles de routage de type état de lien (EL) (*link state* en anglais) sont parmi les plus utilisés dans les réseaux IP actuels [Iyer et al.03], leur simplicité et extensibilité ayant été essentielle à la croissance d'Internet [BA et al.01]. Cependant, avec l'évolution de ce dernier, il y a eu une croissance de la demande de Qualité de Service (QdS) offerte par des réseaux, ce qui a mis en valeur les limitations inhérentes des protocoles de type EL. Dans notre recherche, nous nous concentrons sur le problème de congestion causé par l'utilisation de chemin à coût minimal (CCM). Le problème de congestion dû au CCM est une source de pertes de paquets dans les routeurs, et augmente le délai dans les files d'attentes. Ce délai est une cause importante de gigue (variation du délai) subie par les paquets traversant le réseau [Wischik et al.05]. Notre projet de recherche analyse le Coroutage, qui est une méthode hybride combinant deux protocoles de routage: l'approche classique avec algorithme de CCM, et l'approche de routage explicite, dans laquelle les paquets de données ont leurs routes définies à la source.

Nous allons démontrer que le Coroutage est capable à la fois de réduire la pression sur les liens congestionnés d'un réseau, ainsi que d'améliorer la gigue et le délai pour les flots de type temps-réel. En effet, nous estimons que les applications de type temps-réel, telles que la voix et la vidéo sur IP, requièrent une attention particulière. Ce sont les applications qui sont le plus affectées par la gigue. Notre objectif est donc d'étudier comment améliorer la performance d'un réseau de communication à commutation de paquets, avec une concentration particulière sur l'amélioration des flots temps-réel. Nous planifions d'atteindre nos objectifs en faisant face à la question de la distribution déséquilibrée du traffic due au paradigme de routage CCM à chemin unique. Comme la majorité des flots temps-réel utilisent le protocole de transport UDP, nous avons analysé les caractéristiques de UDP versus celles

de TCP, et avons conclu que ces deux types de flots se comporteraient mieux s'ils étaient séparés. Notre approche vise à permettre aux flots UDP d'éviter les quelques liens congestionnés en les routant de façon explicite. Les flots TCP seront routés selon l'algorithme de CCM, et donc il y aura un seul chemin pour chaque paire O/D, alors que les flots UDP seront routés explicitement sur des chemins composés de liens sous-utilisés.

Nous débutons notre analyse avec une introduction de notions importantes, suivie d'une description générale du concept du Coroutage. Ensuite, nous ferons une revue de la littérature résumant l'état de l'art dans le domaine des protocoles de routage et de la QdS. En particulier, nous allons décrire plusieurs concepts de Christophe Proust, auteur d'un *Internet Draft* intitulé *Routing at the Flow Level* [Proust04]. L'algorithme de routage explicite dans le Coroutage est inspiré d'idées du texte de Proust.

L'étape qui suit consiste à spécifier le Coroutage et ses différents mécanismes. Nous analysons aussi pourquoi la stratégie de séparation de trafic choisie est TCPvsUDP. Des résultats expérimentaux sont ensuites obtenus et analysés à l'aide du simulateur de réseaux NS2. Une analyse comparative est utilisée entre la performance du routage dit standard et entre celle du Coroutage. Le terme routage standard sera utilisé pour le reste de ce texte, et se réfèrera au protocole de routage OSPF, qui utilise un seul CCM par O/D.

CHAPITRE 1

LE CONCEPT DU COROUTAGE

1.1 Définitions de base

1.1.1 Réseaux et graphes

Un réseau de communication est composé de commutateurs et de liens. Les deux types principaux de réseaux de communications sont les réseaux à commutation de circuits, tel que le réseau téléphonique terrestre, et à commutation de paquets, tel que le réseau Internet. C'est la commutation de paquets qui nous intéresse, en particulier les réseaux utilisant le protocole IP (pour *Internet Protocol*), et qui sont parmi les plus répandus actuellement.

Les commutateurs de paquets sont communément appelés *routeurs*. Ce sont les routeurs qui reçoivent des paquets, et qui décident vers quel routeur voisin les envoyer. Une façon commune de représenter un réseau IP est d'utiliser un graphe, avec les nœuds représentant les routeurs, et les arêtes représentant les liens de communications. Voir par exemple la figure 1.1. Les liens de communications sont caractérisés par leur débit maximum de transmission, appelé aussi *bande passante*. La bande passante est généralement mesurée en MégaOctets par secondes (ou MB/s pour *MegaBytes per second*), ou en GigaOctets/s (GB/s).

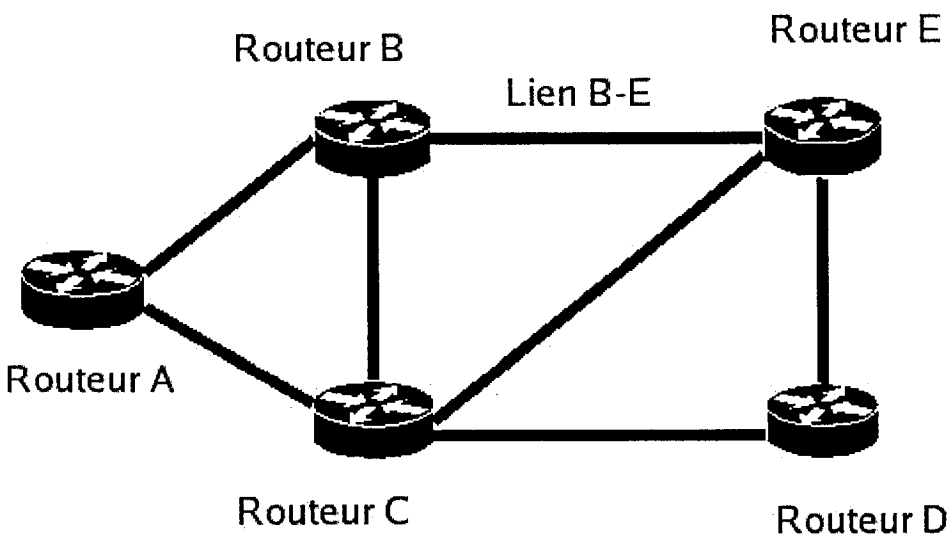


Figure 1.1 Réseau de communication

1.1.2 Les protocoles de routage

Lorsque de l'information doit être transmise d'un usager à un autre, un chemin doit être emprunté par les paquets. Ce chemin est en fait une succession de routeurs, chacun transmettant un paquet vers un routeur qui lui est directment connecté. Il est donc nécessaire d'avoir un protocole de routage déployé dans tout réseau de communication à commutation de paquets. En effet, un mécanisme doit exister pour décider quelle séquence de routeurs un paquet ou un flot doit emprunter.

L'approche générale pour trouver le chemin convenable est de traiter un réseau comme un graphe et d'affecter un certain coût à chaque lien de ce réseau. Le coût peut représenter une ou plusieurs contraintes par exemple, le délai de propagation ou la bande passante du lien. L'allocation de ces coûts dépend de l'administrateur du réseau en question, qui décide de la métrique qu'il désire utiliser. Une métrique de coûts fréquemment utilisée est une approche recommandée par les manufacturiers de routeurs Cisco, et qui consiste à assigner un coût inversement proportionnel

à la capacité du lien [Fortz et al.00].

Une fois les coûts alloués, trouver le chemin optimal qui doit être emprunté est en général analysé comme un problème de théorie des graphes, et résolu avec des algorithmes de calcul de chemin optimal. Un exemple connu est l'algorithme de calcul de chemin à coût minimal de Dijkstra.

À part la notion de déterminer les métriques de routage, il faut aussi décider qui dans le réseau va calculer les routes. Les deux prochaines sections apportent une explication générale à ce sujet.

1.1.2.1 Les protocoles de routage distribués

Avec un protocole de routage de type distribué, chaque routeur est responsable de déterminer le prochain routeur pour un paquet entrant. En d'autres termes, pour tout paquet entrant dans le réseau, la route que va suivre ce paquet n'est pas déterminée à l'avance, mais est trouvée au fur et à mesure que ce paquet traverse les routeurs. Par exemple, les protocoles dits à état de lien (protocoles EL), sur lesquels nous allons revenir plus en détail, procèdent tel qu'illustré dans la figure 1.2. À son entrée, le paquet IP est analysé par le premier routeur, qui détermine le chemin à coût minimal vers la destination. Ayant ce chemin, le routeur sait à ce moment vers lequel des commutateurs qui lui sont directement reliés envoyer le paquet. Dans la figure 1.2, le chemin à coût minimal pour l'origine/destination (O/D) A-D est A-C-E-D. Les routeurs A, C, E effectuent un calcul de CCM vers D et déterminent en conséquence que leurs prochains sauts respectifs sont C, E et D.

Ainsi, dans un protocole de routage distribué, les chemins suivis par les données sont déterminés par tous les routeurs concernés. Une telle approche présente plusieurs avantages. D'abord, le réseau est plus robuste, puisqu'aucun routeur

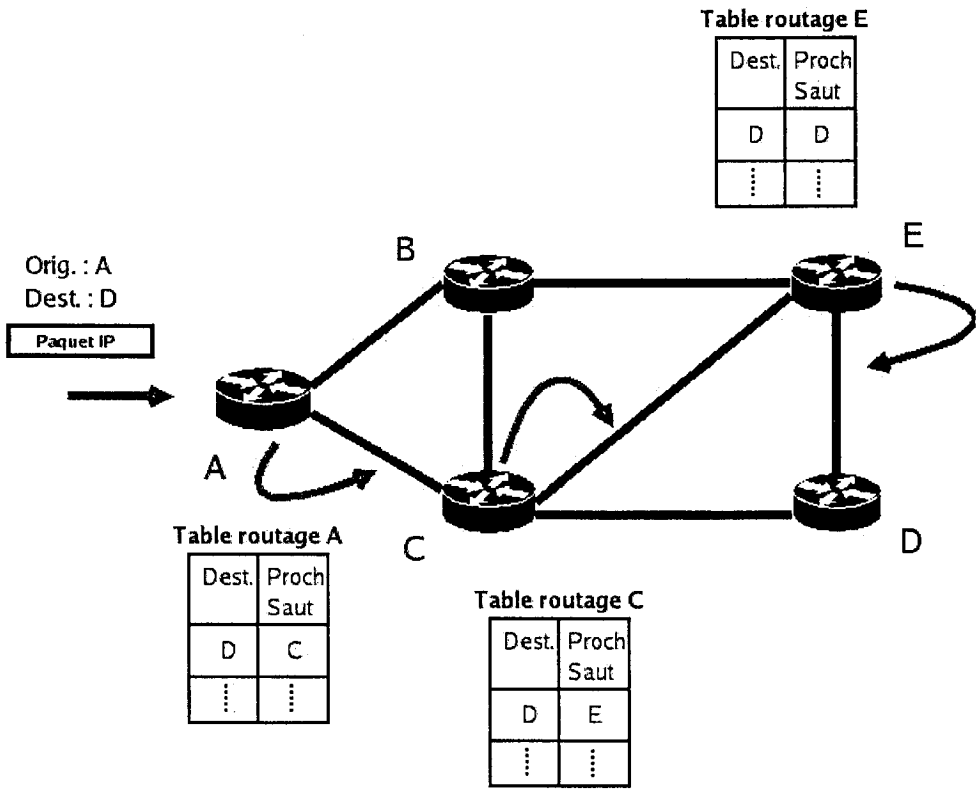


Figure 1.2 Exemple de protocole de routage distribué

n'est indispensable au bon fonctionnement du réseau. En effet, un arrêt de fonctionnement est une information qui est distribuée dans le reste du réseau, ce qui fait qu'une ou plusieurs pannes de commutateurs ne paralysera pas le réseau. Ensuite, puisqu'un routeur est indépendant des autres, il doit faire un analyse complète afin de déterminer le prochain saut pour un paquet, ce qui permet de prendre en considération les conditions locales du réseau. Par exemple, si un lien tombe en panne, les routeurs distants n'ont pas encore eu le temps de recevoir cette information, mais le routeur qui est directement connecté à ce lien est immédiatement conscient de cette situation, et évitera d'y envoyer des paquets.

1.1.2.2 Les protocoles de routage centralisés

Par opposition aux protocoles distribués de routages, les protocoles centralisés effectuent une analyse complète, et à l'avance, pour un ou plusieurs paquets de données. Au moment de la transmission des données, les décisions de routage sont déjà prises. Ainsi, les routeurs intermédiaires jouent un rôle moins important dans une telle situation, ce qui implique que nous aurons l'avantage d'un temps de traitement réduit dans les routeurs intermédiaires. Aussi, on peut exercer potentiellemnt un plus grand contôle sur le chemin suivi par les données, puisque la prise de décision se fait en une fois.

Le routage explicite est un cas particulier de routage centralisé. Il permet de déterminer entièrement la séquence de routeurs que doivent suivre les données, avant que ces dernières ne commencent à être transmises. Le chemin en tant que tel est soit encodé dans les paquets IP eux-mêmes, soit copié dans les routeurs formant le chemin. Dans le second cas, les routeurs intermédiaires enregistrent l'état du flot le traversant: l'identité du flot, son origine et sa destination, son chemin, ainsi que d'autres paramètres sont mémorisés dans les routeurs. Cet ensemble d'informations est référé par le terme état de flot (*flow state* en anglais). Nous verrons plus tard que créer et maintenir beaucoup d'états de flots dans un réseau est une charge, en terme de mémoire et de signalisation, qu'il faut essayer de limiter. La figure 1.3 illustre un protocole de routage explicite simplifié, dans lequel les routes sont encodées dans les paquets mêmes.

1.1.3 Les flots de données

Un réseau permet à ses différents routeurs de communiquer entre eux, et aussi de communiquer avec d'autres réseaux. L'ensemble des informations traversant un

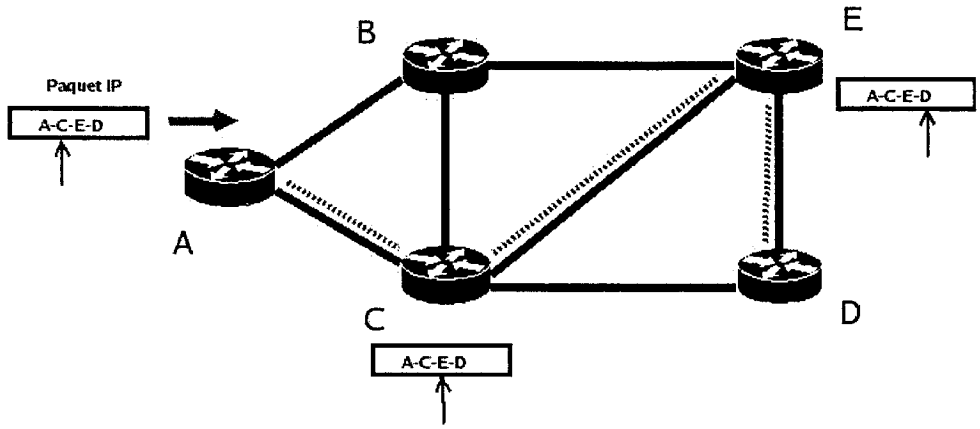


Figure 1.3 Exemple de protocole de routage centralisé

réseau est appelé le trafic de données. Un ensemble de données faisant partie d'une même application est appelé un flot de données. Pour un flot, nous pouvons soit avoir une communication orientée connexion ou non, dépendamment du protocole de transport utilisé. Dans un premier cas, le nœud origine peut tout simplement envoyer les paquets de données l'un à la suite de l'autre, sans s'occuper de savoir si les paquets transmis sont bien arrivés à destination. Dans le second cas, le protocole de transport s'assure que les paquets arrivent bien à leur destination, et retransmet ceux qui sont éventuellement perdus en chemin.

La première approche est utilisée par le protocole de transport UDP (*User Datagram Protocol*), et la deuxième est utilisée par le protocole TCP (*Transport Control Protocol*). Ainsi, UDP et TCP représentent les deux manières fondamentales de transporter des paquets d'une origine à une destination dans un réseau de communication. UDP utilise une approche sans connexion, avec laquelle il n'est pas possible de savoir ce qui est arrivé aux paquets une fois qu'ils ont quitté un routeur. Le protocole UDP est donc simple, et envoie des données sans considérations de fiabilité ou de l'état du réseau. Il est surtout utilisé pour les applications temps-réel, telles que la voix sur IP ou la vidéo sur IP, puisqu'elles peuvent perdre quelques

paquets en chemin sans que l'utilisateur à la destination n'en soit affecté.

D'un autre côté, le protocole TCP établi une connexion entre le nœud origine et le nœud destination. Le nœud origine vérifie ainsi, à l'aide d'accusés de réception envoyés par la destination, si les paquets sont bien arrivés. Si un évènement quelconque fait en sorte qu'un paquet de ce flot TCP se perd en chemin, le routeur d'entrée s'en rendra compte et le transmettra à nouveau. Le protocole TCP est donc fiable, mais requiert l'établissement d'une connexion et d'un système de feedback pour les paquets perdus. En revanche, TCP permet de varier son taux d'envoi de paquets en fonction de la perte de paquets ressentie. Ainsi, TCP prend en considération l'état présent du réseau, tel que la congestion de liens du réseau due à un trafic de données trop large.

Note approche de routage vise à améliorer la performance dans les réseaux en misant sur la différence entre les caractéristiques des protocoles TCP et UDP. Les mécanismes dans le Coroutage ont été conçus avec ces différences en tête. Nous verrons dans une prochaine section les détails des caractéristiques TCP et UDP qui nous intéressent, et comment nous en tirons parti.

1.2 Le Coroutage: objectif et définition générale

1.2.1 Objectif du Coroutage

Le Coroutage priorise les applications de voix et de vidéo sur IP, qui constituent les flots dits temps-réel et qui utilisent en majorité le protocole UDP. Notre objectif est d'améliorer leurs performances en visant spécifiquement ce que nous considérons être le problème principal, celui de la congestion persistente présente dans les réseaux actuels. En effet, nous allons voir en détail que la prédominance des

algorithmes de chemin à coût minimal crée un déséquilibre dans la répartition du trafic dans un réseau. En d'autre termes, quelques liens du réseau en question se trouvent chargés de la majeure partie du trafic, alors que les autres liens transmettent peu ou pas de données.

Nous comptons démontrer que le problème de congestion persistente dû aux algorithmes de chemins à coût minimal est un obstacle majeur à une meilleure performance des flots temps-réel. Aussi, nous comptons résoudre ce problème en routant les flots UDP différemment des flots TCP. Notre stratégie de séparation de trafic est donc fonction du protocole de transport, et chaque protocole de tansport se verra traité par une approche de routage différente, d'où le terme Coroutage.

1.2.2 Définition

Le Coroutage est constitué de deux approches de routage, explicite et par la méthode classique des chemins à coût minimal, ou méthode classique CCM. Tous les flots UDP seront routés explicitement, et tous les flots TCP seront routés sur des chemins trouvés selon l'algorithme de Dijkstra de CCM.

Les routes explicites des flots UDP éviteront les liens sujets à une congestion persistente, et emprunteront les liens sous-utilisés. Il faut donc que les routeurs aient la capacité de détecter le taux d'utilisation en bande passante de chacun de leurs liens, et de distribuer cette information aux autres routeurs, permettant ainsi au protocole de routage explicite d'éviter les liens sur-utilisés.

Les flots TCP, quant à eux, ne verront aucun changement en terme de routage par rapport à un protocole de routage classique. Ils suivront donc le chemin à coût minimal entre chaque origine et destination. Les mécanismes du Coroutage sont schématisés dans la figure 1.4. Dans l'exemple, le taux d'utilisation de chaque lien

est montré. Les flots TCP et UDP veulent tous deux aller de A à D. Le flot TCP suit le chemin à coût minimal A-C-E-D. Quant au flot UDP, la route explicite qui lui sera atribuée à son entrée évitera le lien C-E parce qu'il est considéré comme congestionné. Nous verrons dans une section ultérieure pourquoi nous avons choisi 50% de taux d'utilisation de lien comme indicateur de sous ou sur-utilisation.

Dans le prochain chapitre, nous allons parcourir la littérature relative à notre sujet, ce qui nous permettra de justifier les choix de conception dans le Coroutage, que nous détaillerons dans le chapitre qui suivra.

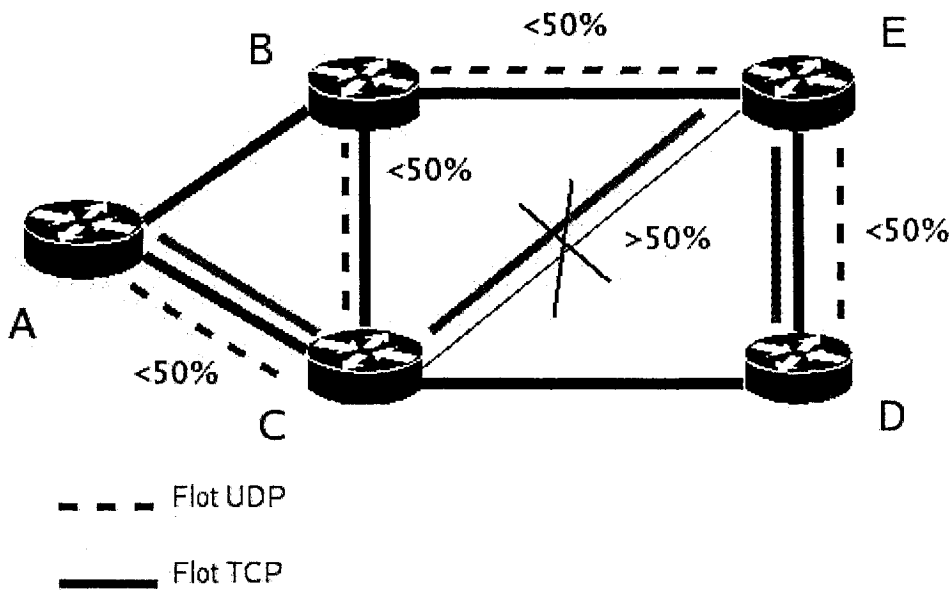


Figure 1.4 Schéma du Coroutage

CHAPITRE 2

REVUE DE LITTÉRATURE

2.1 Les protocoles de routage

Tout réseau à commutation de paquets a besoin d'un protocole de routage. Puisque l'objectif même du réseau est de pouvoir transmettre de l'information entre deux points A et B quelconques appartenant au réseau, il faut déterminer le ou les chemins que doivent emprunter les paquets de données entre A et B. Les fonctionnalités qui permettent la détermination des chemins de données sont regroupées dans un ensemble de règles appelé protocole de routage.

Dans le réseau Internet, le principal protocole de transport est TCP, et le principal protocole de couche 3 est IP. Le souci de la couche 3, dite couche réseau, est d'offrir une méthode d'addressage qui permet d'identifier de façon précise les machines qui sont connectées à un réseau. Grâce à l'unicité des addresses IP, un hôte quelconque dans le réseau peut communiquer avec n'importe quel autre hôte en utilisant son addresse IP. Cette méthode d'addressage a été intentionellement conçu de façon simple afin d'offrir une couche qui serait compatible avec le plus grand nombre de couches liaisons. En d'autres termes, des réseaux utilisant la technologie Ethernet, fibre optique, relai de trame, ou autre, pourront communiquer entre eux grâce à la présence de la couche IP.

L'addressage IP permet donc de faire abstraction de la technologie des couches inférieures. La contrainte permettant de satisfaire le plus de monde était de simplifier IP autant que possible, ce qui a fait que la couche IP possède très peu de

fonctionnalités. Les protocoles de routage, conçus pour router le trafic en suivant l'addressage IP, étaient donc assez simples, et se contentaient de trouver le plus court chemin entre chaque origine et destination. La prochaine section va détailler les protocoles dits des états de liens (*link state* en anglais), qui sont les protocoles de routage les plus répandus dans le réseau Internet.

2.1.1 Les protocoles à état de lien

Les principaux protocoles implémentés actuellement dans Internet sont les protocoles de type état de lien tels IS-IS et OSPF [Iyer et al.03]. Ils opèrent de la façon suivante: tous les routeurs dans un réseau distribuent régulièrement l'état de leurs liens les uns aux autres. Parmi les informations distribuées se trouve le coût de chaque lien, ainsi que le statut du lien (en panne ou non). Ces distributions sont nommées *Link State Advertisements* ou LSA. Le but des protocoles de type EL est de permettre à un routeur de trouver le prochain saut (PS) pour tout paquet le traversant, c'est-à-dire de savoir vers quel routeur transmettre le paquet, et donc quel interface de sortie utiliser. Pour effectuer cette tâche, les routeurs utilisent des algorithmes comme celui du plus court chemin de Dijkstra, pour trouver le chemin à coût minimal, ou CCM, de lui-même vers les autres nœuds du réseau. Une fois le chemin vers la destination déterminé, le PS sera le premier nœud sur le chemin à partir du routeur courant. Quand un paquet IP arrive à un routeur, ce dernier examine l'addresse IP de destination, présente dans l'en-tête du paquet, cherche et détermine d'après sa table de routage l'interface de sortie à travers laquelle il doit transmettre ledit paquet.

L'expression "chemin à coût minimal" est plus exacte que l'expression "plus court chemin", qui est communément utilisée, parce que les coûts sur les liens ne sont pas nécessairement tous égaux à 1. Ceci implique que le chemin trouvé par l'algorithme

de Dijkstra n'est pas nécessairement le chemin le plus court en terme de nombre de nœuds.

2.2 Évolution des demandes en Qualité de Service

Les protocoles de routages OSPF et IS-IS sont relativement simples et anciens. Ils ont été créés dans un souci d'offrir une connectivité logique entre tous les routeurs d'un réseau. Pendant plusieurs années, les besoins des utilisateurs d'Internet étaient assez réduits, et les protocoles EL étaient capables de les combler. Cependant, avec l'avènement d'appplications multimédias, comme la voix ou la vidéo, les utilisateurs se sont mis à demander des performances accrues de la part des réseaux, et des protocoles de routage. En effet, les applications de voix et de vidéo sont dites applications temps-réel, c'est-à-dire que la qualité de la réception des données, telle que perçue par l'utilisateur à la destination, est fortement dépendante du rythme d'arrivée des paquets formant le message. Ainsi, alors qu'auparavant il était simplement nécessaire de s'assurer que tous les paquets arriveraient éventuellement à destination, il était devenu maintenant nécessaire de prendre en considération de nouveaux critères de performance comme le délai inter-paquet ou la variation du délai, aussi appelée gigue.

Avec ces nouvelles demandes, le terme Qualité de Service ou QdS (en anglais *Quality of Service* ou QoS) est apparu, et beaucoup de chercheurs se sont penchés sur le sujet. Le domaine de recherche sur lequel nous nous concentrons est le champ des protocoles de routage dans les grands réseaux, et comment offrir une performance et un service accrus aux données les traversant.

2.3 Mécanismes qui permettent d'offrir la QdS

2.3.1 Intserv versus Diffserv

Il convient d'abord de mentionner *Integrated Services*, ou Intserv. Cette technique de réservation de resources permettait de définir des sessions de flots avec des contraintes très précises, et de réserver la bande passante pour garantir la performance requise. On a rapidement réalisé que Intserv était trop complexe, car il faut maintenir trop d'informations sur les états des flots (*flow states*) dans les routeurs de cœur du réseau. C'est l'exemple typique de la méthode trop lourde en signalisation et en utilisation de resources pour pouvoir être pratiquement implantée dans de grands réseaux. En réaction à l'excès de zèle d'Intserv, la technologie Diffserv est apparue comme étant une approche plus pratique. Diffserv se propose de classifier les flots dès leur entrée dans le réseau et selon leur priorité. Une fois cette classification faite, les paquets de chaque flot sont traités dans les routeurs cœurs en fonction de leurs priorités relatives. Par exemple, un paquet transportant de la voix aurait une plus grande priorité qu'un paquet transportant des données Web, et si ces deux paquets arrivaient en même temps à un routeur, le paquet de voix serait traité avec une plus haute priorité. Ainsi, il n'y a pas à maintenir les états des flots dans le cœur du réseau, réduisant la quantité de signalisation et de resources nécessaires. Dans notre approche, nous essayons autant que possible de réduire la complexité des mécanismes du Coroutage, dans un souci de simplicité et d'extensibilité.

2.3.2 MPLS

La technologie MPLS [Rosen01], ou *MultiProtocol Label Switching*, a été élaborée avec comme objectif de pouvoir combiner la flexibilité de la couche liaison avec le routage logique IP de la couche réseau. L'idée était de pouvoir faire de l'ingénierie de trafic, et donc d'avoir plus de contrôle sur les chemins suivis par les flots et les paquets de données. Avec MPLS, il est possible d'avoir plusieurs chemins possibles pour toute paire O/D. Cette multiplicité de routes par O/D est très limitée dans les procotoles actuels de routage. L'importance de la flexibilité dans le choix des chemins suivis par les flots de données est un aspect important sur lequel nous reviendrons plus en détail, et qui a été pris en considération dans le Coroutage.

2.3.3 Le routage explicite

Dans un désir de contrôler le plus possible l'allocation des ressources aux flots de données, certaines méthodes permettent de router explicitement un flot ou groupe de flots de données. Nous rappelons que dans le routage EL, chaque routeur examine l'addresse IP de destination d'un paquet qui le traverse, et prend une décision indépendante de routage. Au lieu de cette approche distribuée, les méthodes de routage explicite (*explicit routing* ou *source routing* en anglais) effectuent des calculs de CCM au nœud source. Ensuite, la route explicite est enregistrée, dans les paquets ou les routeurs, imposant à tous les paquets concernés la séquence de routeurs précalculée. Dans le routage explicite, la route est déjà fixée lorsque le premier paquet d'un nouveau flot commence à être transmis, et est imposée pour toute la durée du flot.

Le routage explicite est possible avec Intserv, de même qu'avec MPLS. Cependant, le coût en ressources et en signalisation de ce genre de contrôle peut être pro-

hibitif pour de grands réseaux. MPLS en particulier a reçu beaucoup d'attention récemment pour ce qui est de la possibilité d'implémentation à grande échelle.

Cependant, vu les contraintes associées avec le routage explicite pur, des publications sont apparues dans la littérature proposant des méthodes dites hybrides. Le point de départ a été la réalisation que seule une partie du trafic total a besoin d'être traité de façon prioritaire, alors que l'autre partie reçoit déjà un service satisfaisant avec les protocoles de routages actuels. Ces méthodes ont donc pris des concepts des protocoles états de lien et des concepts du routage explicite, et trouvé un compromis entre la simplicité du premier et la flexibilité du second. La section suivante détaille quelques unes de ces approches.

2.3.4 Les approches hybrides de routage

Plusieurs articles dans la littérature ont examiné les approches dites hybrides de routage. Elles proposent de diviser le traffic en deux types, selon certains critères, et de les traiter différemment en terme de routage. On peut alors considérer qu'une même topologie physique, composée de routeurs et de liens de communication, se trouvera ainsi séparée en deux réseaux logiques virtuels différents. Les approches résumées ci-dessous utilisent un mélange OSPF/MPLS, où un type de trafic sera routé avec la méthode classique OSPF, tandis que l'autre type de trafic, normalement le trafic prioritaire, se verra routé avec MPLS.

Une première approche est celle de Bagula [Bagula04], qui différencie les flots entrant dans un réseau en terme de leur demande en bande passante. Il utilise les termes *High Bandwidth Demanding (HBD) traffic* et *Low Bandwidth Demanding (LBD) traffic*, considérant les flots temps-réel dans la première catégorie. Ensuite, on trouve le chemin avec le coût minimal, où le coût varie selon le nombre de flots

qui les traversent et la bande passante qui y est déjà réservée. Des contraintes additionelles sont imposées afin de vérifier que tous les liens du chemin peuvent accepter la demande du nouveau flot.

Une deuxième approche, celle de Riedl [Riedl03], consiste à optimiser les poids des liens avec des algorithmes génétiques, afin de minimiser l'utilisation maximum des liens, et ce avec l'aide des LSP de MPLS pour mieux distribuer la charge du trafic total. Il n'y a cependant pas d'objectif d'amélioration du service offert aux flots temps-réel, et la méthode requiert d'une connaissance de la matrice de trafic.

Enfin, l'approche de Pham et Lavery [Pham et al.03], demande aussi une matrice de trafic, et se propose de séparer les flots selon leur taille, c'est à dire les gros flots (les éléphants) et les petits flots (les souris). L'utilisation des liens est d'abord calculée en utilisant juste le routage OSPF. Une fois les liens sur-utilisés identifiés, les gros flots seront routés avec MPLS avec comme objectif d'optimiser le délai moyen bout-à-bout des paquets.

Notre méthode se différencie par le fait qu'elle se spécialise dans la séparation TCP/UDP, basée sur les caractéristiques intrinsèques de ces deux protocoles de transport. De plus, notre méthode est relativement simple, parce que rien ne change pour les flots TCP. En effet, les flots TCP dans le Coroutage sont toujours traités comme ils le sont dans l'Internet meilleur-effort actuel, parce que la partie routage explicite du Coroutage ne vise que les flots UDP, alors que les TCP sont routés par CCM et donc suivent les mêmes routes qu'avec le routage classique.

Avec le Coroutage, et contrairement à certaines méthodes hybrides, il n'est pas nécessaire de faire des calculs *offline* ou d'utiliser une matrice de trafic préalablement estimée, ce qui rend cette méthode particulièrement simple. En plus de la simplicité, nous allons exposer, dans la section suivante, une liste de critères que nous

comptons suivre dans notre conception du Coroutage.

2.4 Critères pour les protocoles de routage dans les réseaux dorsaux

2.4.1 Critère de multiplicité de chemins

Tout protocole de routage cherchant à réduire la congestion doit être capable de trouver et d'établir plusieurs chemins entre n'importe quelle paire O/D. Ceci est un élément fondamental dans l'ingénierie de trafic, et permet d'avoir de la flexibilité puisque l'on n'est plus limité par le paradigme de chemin unique par O/D.

2.4.2 Critère de granularité fine de routage

Pour la robustesse et la stabilité, il faut aussi avoir une granularité fine de routage. La granularité du routage est la plus petite entité que les routeurs reconnaissent et analysent. Par exemple, router au niveau du flot est considéré comme étant d'une granularité fine, parce que la bande passante utilisée par un seul flot est bien inférieure à la capacité d'un lien de réseau dorsal. Par contraste, Fortz et al. [Fortz et al.00] ont proposé une méthode d'optimization du routage dans laquelle les coûts des liens sont séléctionnés de façon appropriée. L'idée est d'augmenter ou de réduire le coût sur un lien selon qu'on veuille plus ou moins de trafic sur ce lien. Cependant, les résultats de plusieurs études ont conclu que l'optimisation en temps réel des coûts des liens est indésirable, simplement à cause des larges quantités potentielles de bande passante qui peuvent être déplacées d'un lien à l'autre, amenant des disruptions et oscillations dans le réseau. En particulier, des résultats obtenus par AT&T [Feldmann et al.00], ainsi que ceux cités dans RFC3272 [Awduche02] mentionnent le fait que router avec une granularité au niveau des liens est trop

lourde pour les grands réseaux de communication. Dans le cas de AT&T, les auteurs mentionnent que "un léger changement (dans les poids OSPF des liens) peut avoir un impact global significatif". Cette analyse confirme notre choix de router à un niveau plus fin que celui des liens. À titre de comparaison, le routage standard (OSPF ou IS-IS) a une granularité au niveau du paquet IP.

2.4.3 Critère de routage au niveau des flots

Le premier critère, la multiplicité des chemins par O/D, crée de nouveaux problèmes qui doivent être considérés. Par exemple, pour les flots prioritaires, comment divise-t-on le trafic d'une paire O/D sur plusieurs chemins sans avoir le problème de paquets arrivant hors d'ordre (*out-of-order*), c.à.d avoir les paquets d'un même flot qui ne suivent pas le même chemin vers la destination? La solution est d'imposer que tous les paquets d'un même flot doivent suivre la même séquence de routeurs. Ceci implique que la granularité de routage doit être au niveau du flot. En Coroutage, les flots UDP sont routés explicitement par flot, alors que les flots TCP sont routés par l'approche classique CCM, et donc par paquet.

2.4.4 Critère d'extensibilité

Lorsqu'on route par flot, la question de l'extensibilité en terme de charge de signalisation apparaît. En effet, les modèles comme Intserv pouvaient router des sessions individuelles, mais la quantité de signalisation nécessaire l'a rendu impraticable pour les grands réseaux puisque les routeurs de cœur du réseau devaient traiter et maintenir en mémoire l'état de chaque session. Par contraste, le routage EL ne maintient aucun état de flot dans les routeurs cœurs, et c'est cette extensibilité qui explique en partie son adoption dans les réseaux dorsaux. Notre opinion est

que tout protocole de routage pour de grands réseaux doit être essentiellement *core stateless*, c'est-à-dire maintenant la moindre quantité possible d'informations sur les données traversant le réseau. D'un autre côté, les routeurs périphériques des réseaux (routeurs *edge* en anglais) ont typiquement une charge de traffic relativement petite, et peuvent analyser les flots, et maintenir des informations d'état par flot. Ceci est le paradigme *core stateless, edge stateful*, utilisé par exemple par DiffServ. Notre proposition vise à suivre ce paradigme: les routeurs d'accès classifient les flots UDP entrants, et vont ajouter dans les en-têtes des paquets assez d'information pour le routage, afin que les routeurs cœurs n'aient pas à identifier à quel flot appartient chaque paquet. Chaque paquet est analysé individuellement, évitant ainsi d'avoir à maintenir les état des flots.

2.4.5 Critère de stabilité

Les actions prises pour contrôler la congestion doivent créer le moins de perturbation dans ce dernier. Par exemple, une approche réactive de résolution de congestion détecterait qu'un lien est saturé, et essaierait en conséquence de rerouter les flots concernés. Cette approche pourrait causer une perte de données et/ou l'arrivée de paquets en désordre lors de la phase transitoire. De plus, elle pourrait potentiellement causer un grand mouvement de bande passante dans le réseau, causant les oscillations et instabilités mentionnées auparavant. Ainsi, tout ajout de méthode de résolution de congestion ne doit pas réduire la performance globale du réseau, ou augmenter la charge de signalisation de manière significative. Notre approche essaye d'éviter la congestion avant qu'elle n'ait lieu: les flots entrants UDP tentent d'éviter les liens qui montrent des signes d'utilisation intensive future.

2.4.6 Critère de simplicité

Finalement, nous citons l'importance de la simplicité, et la considérons comme étant un élément important pour un réseau Internet viable. Dans [Willinger et al.03], on mentionne qu'il faut essayer d'atteindre *la robustesse à travers la simplicité.* Dans cette optique, le Coroutage est conçu pour améliorer la performance générale d'un réseau en accédant aux liens inutilisés ou peu utilisés. Cette méthode ne vise pas à offrir des garanties mathématiques de qualité de service pour chaque flot, et risquer d'imposer une trop grand complexité. Le Coroutage se concentre plutôt sur une amélioration qualitative au réseau, qui en conséquence améliorerait le service offert aux flots le traversant, en particulier aux flots UDP.

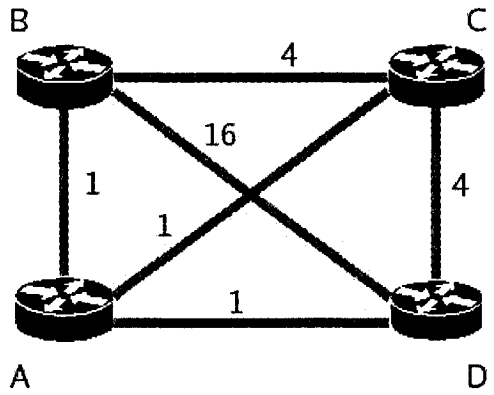
CHAPITRE 3

LA CONCEPTION DU COROUTAGE

3.1 Le problème de congestion dans les réseaux avec protocoles EL

L'utilisation des protocoles EL pose problème dans la mesure où l'agorithme de CCM entraîne un déséquilibre dans la distribution de trafic dans le réseau. En effet, après avoir trouvé le CCM pour une paire de nœuds Origine/Destination, tous les paquets de données composant le traffic de cette O/D vont aller sur cet unique CCM. Il en va de même pour toutes les paires O/D du réseau, chacune utilisant un chemin unique. Les CCM sont trouvés en fonction des coûts affectés à chaque lien par l'administrateur réseau. Pour les routeurs Cisco, on utilise par défaut un coût inversement proportionnel à sa capacité [Fortz et al.00]. En conséquence, les liens ayant la plus grande capacité seront favorisés par les algorithmes de calcul des CCM, alors que les liens à faible capacité auront moins tendance à être choisis.

Cette situation est généralisée dans les réseaux actuels, parce que ces derniers utilisent en majorité les protocoles des états de liens (OSPF ou IS-IS). Ainsi, en pratique, seuls quelques liens de communication vont porter la majorité des données transitant le réseau, alors que les liens restants achemineront relativement peu de paquets [Yilmaz et al.02]. Cette distribution inégale de la charge amène non seulement un gaspillage de bande passante sur les liens sous-utilisés, mais aussi et surtout cause de la congestion dans les liens sur-utilisés. Ces quelques liens "goulot d'étranglement" sont parmi les obstacles principaux face à un réseau robuste et extensible.



Orig.	Dest.	Chemin
A	B	AB
A	C	AC
A	D	AD
B	A	BA
B	C	BA - AC
B	D	BA - AD
C	A	CA
C	B	CA - AB
C	D	CA - AD
D	A	DA
D	B	DA - AB
D	C	DA - AC

Figure 3.1 Exemple du problème de congestion persistente dû à l'algorithme de CCM

La figure 3.1 illustre le phénomène de congestion persistente dont témoignent les réseaux actuels. On utilise une topologie simple à quatre nœuds et 6 liens de communications. Conformément à la méthode par défaut de Cisco, les coûts des liens sont inversement proportionnels à leur capacités. Ainsi, les liens de coût égal à 1, 4 et 16 ont respectivement des capacités égales à 64, 16 et 4 MB/s. Le tableau adjacent à la figure indique l'ensemble des paires O/D possibles, ainsi que le chemin à coût minimal calculé. On remarque que les liens bidirectionnels AB, AC et AD transportent la totalité du trafic, alors que les liens BC, BD et CD sont ignorés. Ainsi, les 12 chemins trouvés pour l'ensemble des paires O/D utilisent juste 3 des 6 liens du réseau.

Nous voyons donc que le problème de congestion persistente est une conséquence de deux choix de conception: les coûts inversement proportionnels aux capacités, et l'utilisation d'un seul chemin. Ce phénomène est une source de pertes de paquets dans les routeurs, ainsi que de délais de file d'attente, ces derniers étant une cause importante de la gigue subie par les paquets traversant le réseau [Wischik et al.05].

Comme notre objectif est d'améliorer la performance des flots UDP temps-réel en combattant la gigue, il nous est apparu logique de nous attaquer à ce problème de congestion.

3.2 Le routage explicite dans le Coroutage

3.2.1 Le routage explicite de C. Proust

Proust [Proust04] a aussi reconnu l'importance de la congestion persistente comme obstacle majeur à une meilleure performance pour les flots temps-réel. Il en a déduit une approche de routage explicite par flot qui prend en considération le taux d'utilisation des liens du réseau, et dont nous nous sommes inspirés pour la conception de la partie routage explicite du Coroutage.

Une première idée était pour les routeurs de pouvoir régulièrement connaître le pourcentage d'utilisation des liens qui leurs sont directement connectés. Chaque routeur enregistre régulièrement la quantité de bande passante utilisée par chacun de ses liens. Ainsi, en divisant la bande passante utilisée par la capacité totale du lien, on obtient le taux d'utilisation du lien, ou taux UL. Les routeurs distribuent ensuite les taux UL de leurs liens et les envoient aux autres routeurs suivant la technique du *flooding* qui est déjà utilisée par les protocoles de routage EL.

Une deuxième idée était de spécifier une valeur seuil de taux UL, au-delà de laquelle un lien est considéré comme sur-utilisé et donc indésirable pour un nouveau flot entrant, et en-dessous de laquelle un lien est considéré comme sous-utilisé et donc capable de router le nouveau flot au besoin.

Nous avons adopté ces deux idées, et allons expliquer dans la prochaine section comment nous les avons utilisées dans l'agorithme de routage explicite du Coroutage.

3.2.2 Spécification du routage explicite du Coroutage

A partir de l'étude faite par Proust, nous allons détailler l'algorithme de routage explicite utilisé dans le Coroutage, sachant qu'il sera seulement utilisé sur des flots UDP temps-réel. D'abord, une nouvelle métrique va être introduite, celle du taux UL. Cette valeur est égale à la bande passante utilisée en moyenne, sur un lien unidirectionnel, divisée par la capacité du lien. On suppose donc que les routeurs peuvent mesurer les taux UL, et que les LSA, *Link State Advertisements*, les informations sur les états de liens distribués entre les routeurs, contiendront la nouvelle métrique. Cette augmentation de la distribution des états de liens a déjà été abordée dans la littérature, par exemple dans [Moy98] et [Awduche01].

Nous allons expliquer maintenant le concept de répandre l'information lorsqu'un lien change de statut et devient soit sous-utilisé, soit sur-utilisé. Un routeur considère qu'un de ses liens devient sur-utilisé lorsqu'il dépasse une valeur prédéfinie de taux UL. Nous avons fixé cette valeur à 50%. Notre choix était arbitraire parce que nous n'avons pas trouvé dans la littérature une étude sur le sujet, et nous voulions avoir une valeur seuil bien inférieure à 100%, pour des raisons que nous verrons plus tard. Ce seuil doit être le même partout dans le réseau. Ainsi, 50% est la limite d'utilisation d'un lien, au-delà de laquelle tous les routeurs du réseau vont considérer le lien comme indésirable pour tout nouveau flot UDP. Lorsqu'un lien dépasse la valeur seuil, son routeur encode le pourcentage UL dans un LSA, et déclenche la procédure qui distribuera le LSA à tous les autres routeurs (*le flooding*). Similairement, lorsqu'un routeur détecte qu'un de ses liens est descendu sous les 50%, la même procédure d'écriture de l'UL et de la distribution du LSA est déclencheée. En conséquence, à n'importe quel moment, les routeurs possèdent, en plus de l'information sur la topologie, les coûts et les statuts des liens, des informations sur quels liens dans le réseau sont considérés comme sur-utilisés, et quels

liens ne le sont pas. Le routage explicite des flots s'ensuit donc de la façon décrite dans le prochain paragraphe.

L'algorithme de routage explicite du Coroutage est une méthode distribuée qui doit donc être déployée dans chaque routeur du réseau. Lorsqu'un nouveau flot entre dans le réseau, le chemin explicite pour ce flot doit être déterminé, et doit être composé d'une séquence de liens qui ne doivent pas être utilisés à plus de 50% de leur capacité. Le calcul du chemin consiste en deux étapes. Premièrement, l'algorithme rejette tous les liens dont l'utilisation a dépassé le seuil de taux UL. Deuxièmement, la topologie examinée étant maintenant composée seulement de liens sous-utilisés (on a donc un sous-graphe du réseau), les routeurs trouvent le chemin à coût minimal en appliquant l'algorithme de Dijkstra, avec les coûts des liens, inversement proportionnels à leur capacité, comme métrique dans le calcul du chemin. Ce chemin sera la route explicite utilisée par le flot, et ne contiendra en conséquence que des liens sous-utilisés.

Nous croyons qu'il est important de ne pas faire calculer la route explicite entièrement par un seul routeur. Au lieu d'avoir le routeur d'entrée d'un flot comme seul décideur pour la détermination de la route, nous préférons que le chemin soit construit routeur par routeur. Ainsi, on pourra réduire les imprécisions dues au temps qu'il faut pour que les informations se propagent dans le réseau, permettant aux routeurs d'incorporer les états de leurs liens les plus récents.

L'idée est d'utiliser un phase d'initialisation: lorsqu'un nouveau flot entre dans le réseau, un paquet de signalisation est formé dans le routeur d'entrée. Ce routeur déterminera la route explicite, tel que vu ci-dessus, et saura donc quel est le prochain routeur dans la route explicite. Le paquet de signalisation sera en conséquence transmis vers ce voisin. Ensuite, ce deuxième routeur déterminera à son tour la route explicite, et donc son prochain saut, et transmettera le paquet

d'initialisation vers le prochain routeur voisin. Ceci est répété jusqu'à ce que le paquet arrive à la destination, après quoi il retourne en suivant le chemin inverse vers l'origine, contenant les informations nécessaires qui permettront de former et d'enregistrer la route explicite du nouveau flot UDP. La figure 3.2 illustre cette phase d'initialisation. Dans cet exemple, le nouveau flot f veux aller de A à F. Le paquet de signalisation quitte A vers le routeur C, puis vers E et enfin F. En chemin, il collecte les informations qui permettent d'expliciter le chemin à suivre pour les paquets de f. Dans ce cas, ce sont les numéros des interfaces de sortie des routeurs qui sont utilisés. Le dernier chapitre de ce mémoire rentrera plus en détails sur les méthodes d'encodage de chemins explicites.

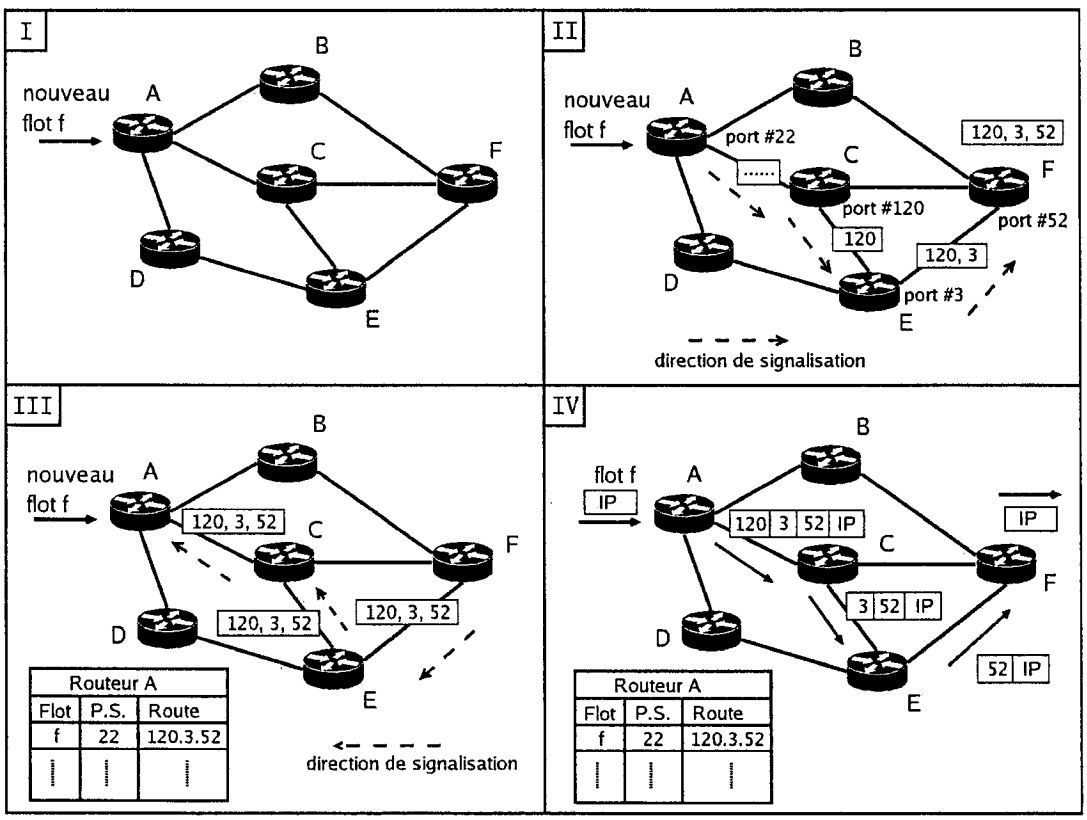


Figure 3.2 Signalisation pour déterminer un chemin explicite

3.2.3 Caractéristiques du routage explicite dans le Coroutage

La phase de signalisation qui vient d'être décrite ci-dessus est similaire aux messages PATH et RESV de la technologie MPLS-TE, dans laquelle un paquet traverse tout le chemin en un aller-retour qui permet de collecter le plus d'informations possibles.

La phase d'initialisation, durant laquelle le chemin explicite se construit, est distribuée entre différents routeurs pour avoir accès aux informations les plus récentes sur les liens. Par la nature d'un routage distribué, il existe des moments d'instabilités durant lesquels l'information n'est pas la même partout dans le réseau. Si le paquet de signalisation se trouve coincé dans une boucle à cause d'une telle instabilité, ce paquet sera éventuellement rejeté du réseau. Dans ce cas, une approche serait de redéclencher la phase d'initialisation. Une autre serait d'abandonner la tentative, et d'envoyer le flot UDP en question sur la route nominale (avec les TCP) par défaut lorsque quelque chose ne marche pas. Cette dernière proposition est celle que nous suivons dans l'intérêt de maintenir la simplicité dans le réseau, ainsi qu'une connectivité de base.

Dans les approches de routage qui utilisent les mesures de bande passante des liens pour choisir les routes, il y a un compromis à faire entre la précision des routes trouvées, dépendante de la précision de l'information disponible, et la quantité de signalisation des LSA dans le réseau. L'exemple typique est l'algorithme dit *Widest-Shortest Path* [Yilmaz et al.02], dans lequel, s'il y a des chemins de même coût, celui avec la plus large bande passante est choisi. Dans le cas du routage explicite dans le Coroutage, un routeur n'a pas besoin de savoir à un certain moment si un lien est à 25, 35 ou 90% UL. La seule chose à savoir est si le lien est sur-utilisé ou non. Ainsi, le calcul des chemins explicites est plus simple et requiert moins de signalisation.

Avoir un seuil d'utilisation à 50%, loin de 100%, permet d'éviter autant que possible le reroutage de flots préexistants. Si un lien contenant des flots UDP passe au-dessus des 50% d'utilisation, cet évènement est distribué par LSA à travers le réseau, assurant ainsi que tous les routeurs ne considèrent pas ce lien comme candidat pour accueillir de nouveaux flots UDP. Cependant, ce changement n'affecte pas les flots UDP déjà sur le lien. En effet, puisqu'un lien qui vient d'être marqué comme sur-utilisé n'est probablement pas bien au-dessus de 50%, les flots qui l'utilisent ne subissent pas de congestion, et n'ont donc pas besoin d'être reroutés. En conséquence, avoir un lien passer d'un état à l'autre ne crée pas de perturbation majeure dans le réseau, dans le sens qu'on n'aura pas besoin de rerouter les flots UDP déjà établis.

La fréquence du *flooding* des LSA causé par des liens traversant la valeur seuil n'est pas élevée. D'abord, le pourcentage UL est mesuré comme une moyenne à long terme et ignore les pics de bande passante du court terme, et donc les rafales ne causeront pas l'émission de LSA. Ensuite, tel qu'expliqué auparavant, le paradigme du chemin unique à coût minimal implique qu'il n'y a pas tellement de liens lourdement congestionnés [Iyer et al.03]. Ces deux éléments suggèrent que le Coroutage ne devrait pas surcharger le réseau avec de la signalisation excessive.

Plusieurs concepts présentés par Proust ont été utilisés pour le routage explicite du Coroutage. Cependant, le Coroutage ne route pas explicitement tous les flots du réseau également, alors que l'approche de Proust route explicitement tous les types de trafic. Nous prenons en considération le large éventail de types de flots composant le traffic Internet, et les caractéristiques spécifiques de certains flots. Dans le Coroutage, seuls les flots UDP temps-réel, qui bénéficieraient le plus d'une réduction de délai et de gigue, seront routés de façon explicite.

3.3 Spécification du Coroutage

Le Coroutage est une approche de routage hybride et distribuée, dans laquelle les flots TCP et les flots UDP sont traités de façon différente. Les paquets entrants dans le réseau sont classés à la périphérie comme étant TCP ou UDP, et traités en conséquence. Puisque l'algorithme de routage explicite est appliqué seulement aux flots UDP, ceci veut dire que TCP sera toujours traité de la façon classique, c.à.d. que les paquets TCP seront routés de la même manière que celle du réseau Internet actuel utilisant OSPF ou IS-IS: lorsqu'un paquet TCP entre dans le réseau, le routeur le transmet vers le prochain saut suivant l'algorithme de chemin à coût minimal de Dijkstra. L'information additionnelle sur les pourcentages UL, et sur quels liens sont sous ou sur-utilisés, n'est *pas* utilisée pour les paquets TCP. Ainsi, les TCP vont aller à travers une seule route, le chemin nominal (ou chemin à coût minimal), exactement comme ils le font dans les réseaux actuels. Nous spécifions le fait que les coûts des liens utilisés dans nos simulations sont ceux utilisés par défaut par les routeurs Cisco.

Quants aux flots UDP, ils sont routés explicitement: lorsqu'un flot UDP entre dans le réseau, l'algorithme de routage explicite détermine la séquence de prochains sauts que tous les paquets de ce flot doivent suivre. Cette route est trouvée grâce à la méthode décrite dans la section 3.2. Ensuite, l'information nécessaire spécifiant la route est enregistrée, soit dans chaque paquet du flot avant qu'ils ne soient transmis vers le prochain routeur, soit dans les routeurs eux-mêmes, selon la méthode d'encodage de route explicite utilisée. Ceci implique que les routeurs d'entrées maintiennent une liste des flots UDP entrants, et peuvent ainsi identifier à quel flot appartient un paquet, et comment le traiter en conséquence. C'est ce genre d'information qui forme l'état d'un flot (ou *flow states*). En faisant en sorte que les routeurs d'entrées maintiennent le gros des informations sur les flots, nous suivons

le paradigme *edge-stateful, core-stateless* expliqué auparavant. Dans la prochaine section, le choix de la stratégie de séparation de trafic est expliqué, et nous montrerons pourquoi nous pensons que TCP et UDP doivent être traités différemment.

CHAPITRE 4

STRATÉGIE DE SÉPARATION DE TRAFIC: TCP VS. UDP

Le trafic actuel dans Internet est dominé par les flots utilisant TCP comme protocole de transport. D'un autre côté, le traffic temps-réel utilise UDP en majorité, et va témoigner d'une forte croissance avec la popularisation d'applications gratuites telles Skype, et l'entrée des opérateurs de télécommunications dans le marché de la voix sur IP. Bien que les flots TCP resteront en toute probabilité les flots dominants, la présence accrue de UDP, avec leur manque de réactivité face à la congestion, devra être gérée attentivement afin de maintenir la résilience et la performance de base d'Internet. Le Coroutage est ainsi vu comme une approche pour gérer de manière simple et efficace ces deux types de flots fondamentalement différents. Nous allons dans les paragraphes suivants présenter plusieurs arguments pour appuyer ce concept.

4.1 Caractéristiques d'ingénierie de trafic

Avec le Coroutage, le trafic TCP suivra la route nominale et aura donc tendance à congestionner les liens qui appartiennent à plusieurs chemins nominaux. Ceci laissera les autres liens peu ou pas utilisés. L'algorithme de routage explicite tentera d'envoyer des flots UDP le long des chemins qui évitent les lien congestionnés, et donc sur des liens peu utilisés par les TCP. Ainsi, le Coroutage contient intrinsèquement des mécanismes d'ingénierie de trafic. Bien que les liens congestionnés ne deviendront pas nécessairement sous-utilisés, parce que les TCP sont toujours largement dominants, l'approche de Coroutage permet d'avoir accès à de

la bande passante inutilisée dans le réseau. L'approche offre une manière flexible d'avoir de multiples chemins entre toute O/D, sans que ces derniers aient à être calculés d'avance, ou à être trouvés selon un algorithme d'optimisation complexe. Les chemins sont créés dynamiquement et seulement selon le besoin.

4.2 L'intéraction entre les flots UDP et TCP

Nous considérons la congestion comme ayant lieu lorsque le taux d'arrivée de paquets, dans l'interface de sortie d'un routeur, approche de la capacité totale de ligne de cet interface. C'est à ce moment que la file d'attente de l'interface commence à se remplir et, si le taux d'arrivée ne diminue pas, il y aura éventuellement perte de paquets. Cependant, même sans cette perte de paquets, le temps passé dans la file d'attente cause de la gigue et du délai supplémentaire. Les paquets TCP sont décrits par Yilmaz et Matta [Yilmaz et al.01] comme étant *bursty and congestion sensitive* (saccadé et sensible à la congestion). Par opposition, les flots UDP sont décrits comme étant *smooth, unresponsive* (moins saccadés et ne réagissent pas face à la congestion) et utilisés pour les flots temps-réel. De plus, les flots TCP peuvent récupérer les paquets perdus, alors que UDP ne le peut pas. Cependant, les flots temps-réel peuvent perdre une petite quantité de paquets sans que l'utilisateur ne perçoive une différence. Enfin, la congestion peut être réduite lorsque les flots TCP réduisent leur fenêtre d'envoi. Ceci implique qu'il y a une inégalité inhérente lorsque des flots TCP et UDP sont en compétition pour les resources d'un lien: Alors que TCP réduirait sa fenêtre d'envoi de paquets (et donc son taux) à la détection de congestion, UDP ne ferait rien de la sorte et bénéficierait de la bande passante ainsi libérée.

Ce dernier point est analysé par [Floyd et al.99], dans lequel les auteurs définissent le terme effondrement dû à la congestion *congestion collapse.* Selon eux, ces

cas d'effondrements sont dûs en majorité au déploiement croissant d'applications n'ayant pas de mécanismes de contrôle de congestion bout-à-bout, c.à.d. les flots UDP. Ces cas se produisent rarement dans le réseau Internet actuel, à cause de la dominance des flots TCP avec leur facultés adaptatives. Les auteurs spécifient que les mécanismes de contrôle de congestion bout-à-bout de TCP ont été facteurs critiques dans la robustesse de l'Internet [Floyd et al.99]. En effet, sur un lien, un agrégat de flots TCP possède une marge de manoeuvre de bande passante, parce qu'il est capable de varier substantiellement son taux d'envoi de données en fonction de la congestion subie, d'où le terme trafic élastique. Floyd et al. concluent que, malgré le fait que les algorithmes de routage actuels causent une répartition inégale de charge de trafic dans un réseau, la prédominance de TCP dans Internet a permis à ce dernier de maintenir sa robustesse durant sa croissance.

Les auteurs ajoutent que, au moins sur les liens qui sont des goulots d'étranglement du réseau, il vaudrait mieux que les flots de type TCP restent largement dominants. Si la croissance du trafic temps-réel continue, le trafic UDP pourrait commencer à prendre une partie non négligeable des ces liens, en supposant qu'on continue d'utiliser un chemin nominal unique par O/D. Ceci réduirait l'élasticité du trafic total sur le lien, et pourrait sérieusement affecter la robustesse et la stabilité générale du réseau Internet.

Le Coroutage traite cette question de façon simple et flexible. En effet, le Coroutage force les flots non coopératifs (UDP) à éviter les liens qui tendent à transporter une lourde charge de trafic, en particulier les quelques liens qui sont sur-utilisés par les algorithmes de chemin de coût minimal. Ainsi, ces lien goulots d'étranglements resteront en large partie dominés par les TCP, ce qui maintiendra la stabilité et robustesse actuelles de l'Internet. Il n'y aura donc pas de dégradation générale de la performance du réseau pendant que les flots UDP continuent à augmenter. On s'assure aussi que les flots TCP ne seront ainsi pas pénalisés pendant que le service

UDP est amélioré.

4.3 La question de la durée des flots

La littérature relative à l'analyse des flots Internet possède une riche nomenclature visant à classifier les flots selon leur taille, leur durée, taux de paquets et forme en rafale. En particulier, Lan et Heidemann [Kc et al.03] mentionnent le fait que 70% des flots Internet durent moins de 10 secondes, et que la majorité des flots sont de type TCP. Cette information nous a poussés à modifier l'approche de routage explicite de Proust, et à ne pas l'appliquer aveuglément à tous les flots. En effet, puisque la plupart des flots ne vont durer que quelques secondes, on peut conclure qu'il n'est pas nécessaire de leur trouver une route explicite. En d'autres mots, si une connexion TCP va être établie entre deux bouts pour être détruite après l'envoi de quelques paquets, alors l'ajout en plus du délai d'établissement de la connexion TCP d'un délai de formation de la route explicite, aurait un affet adverse sur la performance. Puisque ces flots ne sont pas des flots prioritaires, pourquoi augmenter la complexité au nœud d'entrée en forçant ce dernier à classifer chaque flot TCP qui rentre? L'objectif principal est pour le trafic temps-réel de subir moins de gigue en évitant les liens congestionnés qui causent les délais d'attentes dans les files. Cette direction de pensée, que les flots UDP temps-réel sont sensibles à la gigue et durent relativement longtemps, et le fait que les TCP peuvent vivre avec de la congestion et sont en grande partie de courte durée, a influencé notre décision de spécialiser notre algorithme de routage explicite et de ne cibler que les flots UDP.

4.4 Autres stratégies de séparation

On peut envisager d'autres stratégies de séparation de trafic pourraient être considérées, comme par exemple celle de différencier les données au niveau application. Une approche serait de séparer le trafic FTP du reste, et de router explicitement les flots FTP. Cependant, le choix d'application n'est pas évident. Tel que mentionné par Floyd et al., personne ne peut prédire le prochain *killer app*, ou super appplication. De nouvelles applications apparaissent tout le temps, comme par exemple les applications Peer-to-Peer qui actuellement constituent une large partie du trafic Internet. Ainsi, définir une stratégie de séparation de trafic au niveau application n'est pas une garantie de stabilité à long terme. Par contraste, TCP et UDP sont des éléments stables: ils correspondent aux deux manières fondamentales d'établir une communication entre deux points. Ils vont rester sans doute les protocoles de transport dominants, et donc constituent des candidats plus appropriés pour une stratégie de séparation de trafic.

4.5 Les conséquences d'avoir TCP/UDP comme stratégie de séparation

Du côté des UDP, ces flots vont être transmis sur des liens sous utilisés. La bande passante mesurée du lien est seulement un indicateur long terme. La bande passante utilisée varie beaucoup à court terme, d'où le besoin d'une valeur seuil bien inférieure à 100%. Ainsi, il restera une marge de bande passante inutilisée qui absorberait les inévitables rafales de trafic qui ont lieu sur les liens dorsaux. Il est important de préciser le point suivant: le problème de congestion de liens à long terme est résolu en permettant aux flots UDP de les éviter, alors que celui des brusques montées de trafic pour un court temps est résolu grâce à la capacité excédentaire dans les liens utilisés par ces flots UDP.

Le simulateur NS2 est utilisé pour tester si le Coroutage agit selon nos attentes. Tel qu'expliqué dans la prochaine section, une analyse comparative a été utilisée pour juger de la performance du protocole de routage. En effet, pour chaque scénario de simulation, ce dernier est simulé deux fois, en utilisant une fois le Coroutage, et une autre fois le routage OSPF (routage standard).

CHAPITRE 5

SIMULATION

Le logiciel NS2 a été utilisé afin de simuler le Coroutage. Une description de NS2 est disponible en annexe.

5.1 Méthodologie: analyse comparative entre le Coroutage et le routage standard

Pour pouvoir estimer le changement apporté par le Coroutage, nous effectuons une analyse comparative entre le Coroutage et le routage standard de la manière suivante. D'abord, un script de simulation est créé. Il décrit la topologie du réseau (les nœuds, liens, coûts et capacités des liens) et la matrice de trafic (le nombre de flots TCP et UDP ainsi que leurs O/D).

Ensuite, le script en question sera simulé deux fois, une fois avec le routage standard, et une autre fois avec le Coroutage. Cette paire de simulations par script est appelée un scénario. Les fichiers de traces sortant de la simulation avec Coroutage sont alors comparés avec les traces sortantes de la simulation avec routage standard. Comme c'est seulement la méthode de routage qui change entre les deux simulations, et que tous les autres éléments (topologie, trafic, durée de simulation, etc...) sont invariables, les différences entre les traces des deux simulations reflètent donc les différences de comportement des flots et de leur performance causées par l'utilisation de deux méthodes de routage différentes. Une autre façon de le dire est que le routage standard représente la méthode de routage telle

qu'implémentée actuellement dans les réseaux, d'où l'utilisation du terme standard, et en conséquence toute nouvelle approche de routage devra être jugée par rapport à cette méthode standard, et c'est ce que nous faisons avec le Coroutage.

La figure 5.1 illustre notre approche comparative. Lorsque le script est exécuté avec routage standard, il est équivalent à OSPF, c.à.d que *tout* le trafic entre une paire de nœuds O/D suivra un seul chemin (le chemin à coût minimal, que nous appelons route nominale). D'un autre côté, lorsque c'est le Coroutage qui est utilisé, tout le trafic TCP d'une O/D suivra la route nominale, alors que les flots UDP seront explicitement routés le long de chemins non congestionnés, suivant l'agorithme de routage explicite décrit précédemment. Ainsi, rien ne change pour les TCP entre le routage standard et le Coroutage, en terme des chemins suivis. De plus, et dans toutes les simulations, les flots UDP se mettent à transmettre des paquets quelque temps après les TCP, et tous les flots continuent à transmettre des paquets jusqu'à la fin de la simulation.

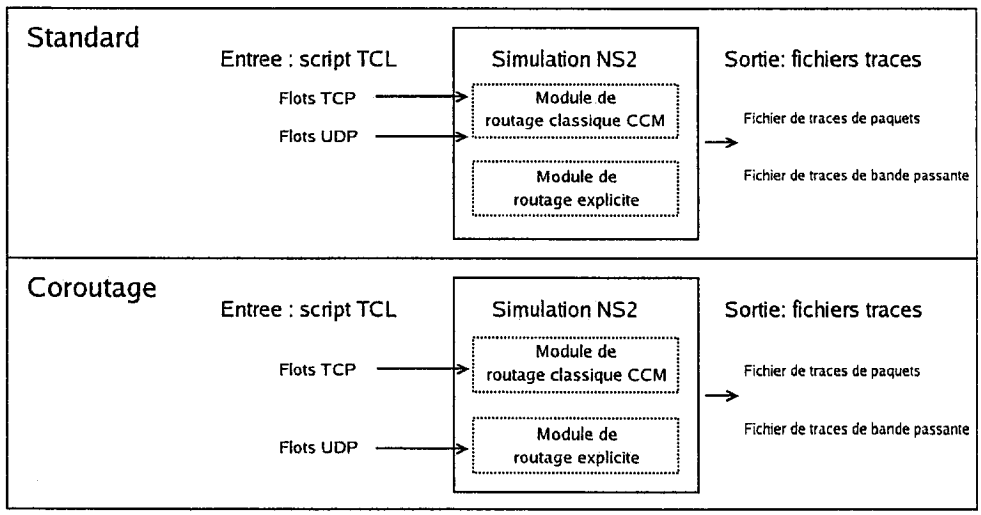


Figure 5.1 Structure des scénarios

5.2 Définitions du délai et de la gigue

Les principaux fichiers de traces contiennent des mesures de délai et de gigue, la gigue étant la variation du délai. Pour l'analyse UDP, le délai de file d'attente bout-à-bout est enregistré pour chaque paquet de chaque flot. Ce délai est égal à la somme des temps passés dans la file d'attente de chaque routeur sur le chemin du paquet en question, à partir de l'entrée du paquet dans le réseau et jusqu'à sa sortie. La gigue pour UDP est la différence (en termes absolus) entre les délais de file d'attente de deux paquets consécutifs du *même* flot UDP. Il est important de mesurer le délai de file d'attente pour les flots UDP, au lieu du délai total bout-à-bout. En effet, on peut avoir plusieurs chemins explicites disjoints entre une paire O/D, et donc différentes valeurs pour les délais de transmission et de propagation. Pour pouvoir juger si une paire O/D a perçu une amélioration de performance avec Coroutage, c'est le délai de file d'attente qui doit être mesuré et comparé lorsque le routage standard puis le Coroutage sont utilisés.

En contrepartie, nous utilisons le délai total bout-à-bout pour les flots TCP, et il est égal à la somme des délais de file d'attente, de propagation et de transmission entre le nœud d'entrée et le nœud de sortie. Puisque tous les flots TCP d'une paire O/D utilisent l'unique chemin nominal de cette paire, en routage standard et en Coroutage, alors tous les paquets de ces flots vont avoir les mêmes délais constants de transmission et de propagation, permettant de faite une analyse comparative entre les deux méthode de routage. En ce qui concerne la gigue TCP, tous les flots TCP d'une même paire O/D sont considérés comme formant un flot aggrégé, et les valeurs de gigue sont calculées en prenant la différence (en valeur absolue) entre les délais bout-à-bout de deux paquets consécutifs de l'aggrégat.

5.3 Topologie

Le réseau étudié est une topologie de 30 nœuds générée aléatoirement, avec un logiciel gratuit disponible sur Internet. Les liens ont des capacités égales à 4, 16 ou 64 Mégaoctets par secondes (MB/s). Les capacités sont des multiples de 4 pour émuler les capacités des fibres optiques. Le coût d'un lien est obtenu en prenant la plus haute valeur (64) et divisant par la capacité du lien en question. Ainsi, les coûts pour 4, 16 et 64 MB/s sont respectivement 16, 4 et 1. Cette méthode est l'approche par défaut utilisée dans les routeurs Cisco pour le calcul des coûts. La topologie est décrite dans la figure 5.2.

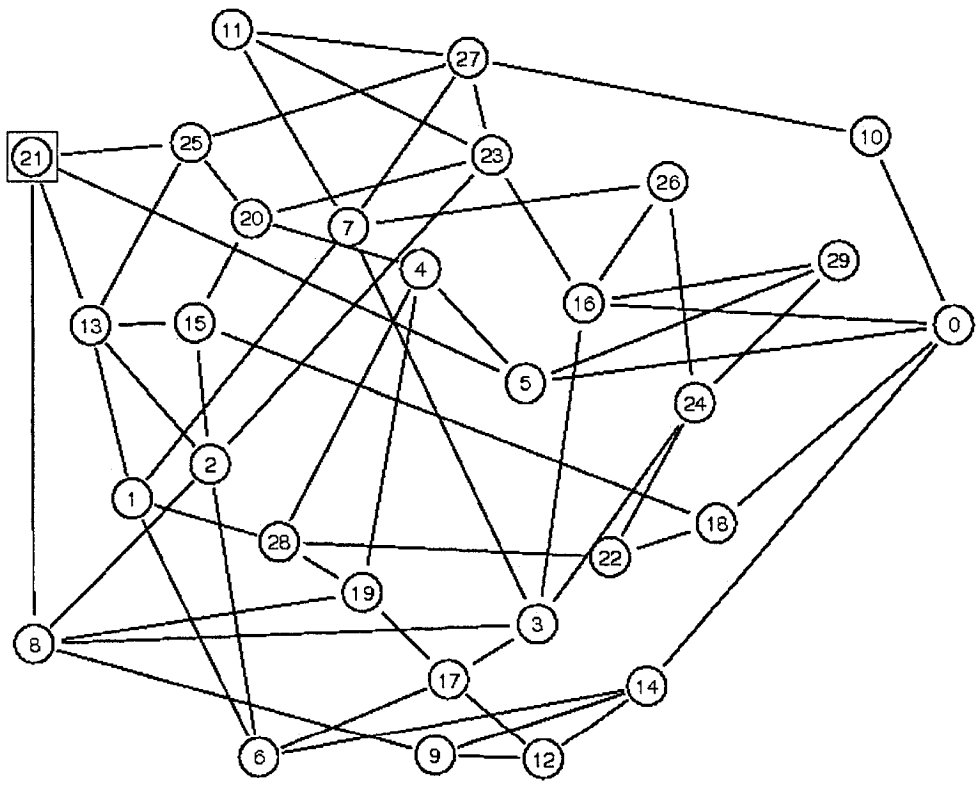


Figure 5.2 Topologie à 30 nœuds générée aléatoirement

Cette topologie est analysée à sept reprises, et donc avec sept scénarios, chacun

Scénario	Nombre de flots TCP et UDP	Caractéristiques
A_1	5000 TCP, 500 UDP	Flots UDP sont à 10% des flots TCP.
A_2	5000 TCP, 500 UDP	L'ensemble des nœuds origine et destination a changé.
A_3	7000 TCP, 700 UDP	Le trafic total a augmenté.
A_4	8000 TCP, 400 UDP	Le trafic total a augmenté. Flots UDP sont à 5% des flots TCP.
A_5	6000 TCP, 3000 UDP	Flots UDP sont à 50% des flots TCP.
A_6	7000 TCP, 4000 UDP	Le trafic total a augmenté.
A_7	8000 TCP, 5000 UDP	Le trafic total a augmenté.

Tableau 5.1 Sommaire de la matrice de trafic pour chaque scénario

étant formé d'une simulation avec routage standard, et une autre avec Coroutage. Les scénarios et leurs caractéristiques sont présentés dans le tableau 5.1, et sont référés par scénario A_x, x allant de 1 à 7.

La deuxième colonne du tableau indique le nombre de flots UDP et TCP formant le trafic total du script définissant le scénario, et la troisième colonne indique les caractéristiques par scénario. En particulier, les caractéristiques pour un certain scénario spécifient ce qui a changé par rapport au scénario qui le précède immédiatement. Ainsi, le scénario A_1 simule un trafic formé de 5000 flots TCP et 500 flots UDP, avec la remarque que le nombre de flots UDP est 10% du nombre de flots TCP. Cette proportion a été utilisée pour refléter le fait que les flots TCP représentent la majorité du trafic Internet actuel.

Le scénario A_2, quant à lui, contient le même nombre de flots que A_1, mais avec la différence que le sous-ensemble de nœuds du réseau qui est utilisé pour les paires Origine/Destination a changé par rapport à A_1. En effet, pour simuler les routeurs d'accès et les routeurs cœurs, seul un sous-groupe des nœuds de la topologie est utilisé pour l'accès. Ce sont à travers ces nœuds seulement que les flots pourront entrer et sortir, les autres nœuds du réseau faisant office de nœuds de transit. Pour

revenir à A_2 et A_1, on a changé le sous-groupe des nœuds O/D pour voir si cela affectera la performance du Coroutage.

Ensuite, le scénario A_3 voit son trafic augmenter à 7000 TCP et 700 UDP. Il en va de même pour A_4, avec la différence que les flots UDP sont maintenant à 5% des flots TCP, au lieu de 10% en A_3. Nous continuons avec A_5, et désirons cette fois que les flots UDP soient à 50% des flots TCP, sans pour autant trop ajouter de charge de trafic par rapport à A_4, ce qui a donné un trafic de 6000 TCP et 3000 UDP. Finalement, A_6 et A_7 ont vu leurs trafics respectifs augmenter, tout en gardant les flots UDP à plus de 50% des flots TCP.

CHAPITRE 6

ANALYSE UDP

Le Coroutage va essayer d'envoyer autant que possible les flots UDP loin des lien congestionnés par les flots TCP. Un premier élément à examiner est donc l'ensemble des liens qui sont utilisés comme routes pour les UDP, et voir ce qui change entre routage standard et Coroutage.

Ainsi, le tableau 6.1 montre comment le routage explicite du Coroutage permet aux flots UDP d'éviter les liens sur-utilisés. Pour chaque scénario et type de routage, le pourcentage de flots UDP traversant au moins un lien utilisé à plus de X% est indiqué, avec X prenant les valeurs 50, 80 ou 95%.

Par exemple, dans le scénario A_3, et avec routage standard, 83.7% de tous les flots UDP ont au moins un des liens sur leurs chemins utilisé à plus de 50%. En Coroutage pour A_3, seuls 6.1% de ces flots UDP vont sur au moins un lien à plus de 50%. L'amélioration entre les deux types de routages est montrée en gras (amélioration de 77.6%), et illustre la capacité de l'algorithme de routage explicite du Coroutage de trouver des routes inutilisées pour une majorité des flots UDP. Ceci est une preuve supplémentaire du déséquilibre de distribution de la charge de trafic qui est causé par le paradigme de l'unique chemin par O/D, de la disponibilité de bande passante libre et de l'existence de routes alternatives.

On peut ainsi clairement voir dans ce tableau que le Coroutage est capable de réduire le nombre de liens surchargés dans les chemins des flots UDP, avec une moyenne d'amélioration de 56.7% , 26.8%, et 13.5% respectivement pour les liens supérieurs à 50, 80 et 95%. En particulier, pour les colonnes des 80 et 95%, le

Scénario		% flots ayant au moins un lien à : plus de 50%	plus de 80%	plus de 95%
A_1	Standard	71.4%	29.4%	0.0%
	Coroutage	20.2%	1.8%	0.0%
	Amélioration	**51.2%**	**27.6%**	**0.0%**
A_2	Standard	51.2%	10.0%	0.0%
	Coroutage	8.2%	2%	0.0%
	Amélioration	**43.0%**	**8.0%**	**0.0%**
A_3	Standard	83.7%	30.1%	11.0%
	Coroutage	6.1%	2.1%	0.0%
	Amélioration	**77.6%**	**28%**	**11.0%**
A_4	Standard	83.3%	39.0%	27.8%
	Coroutage	3.8%	1.8%	1.8%
	Amélioration	**79.5%**	**37.2%**	**26.0%**
A_5	Standard	77.2%	28.4%	11.0%
	Coroutage	18.4%	6.1%	2.7%
	Amélioration	**58.8%**	**22.3%**	**8.3%**
A_6	Standard	86.7%	35.1%	27.1%
	Coroutage	35.5%	2.7%	2.0%
	Amélioration	**51.2%**	**32.4%**	**25.1%**
A_7	Standard	88.0%	36.9%	28.4%
	Coroutage	52.7%	4.5%	4.3%
	Amélioration	**35.3%**	**32.4%**	**24.1%**

Tableau 6.1 Pourcentage de flots UDP traversant au moins un lien qui est utilisé à plus de X% de sa capacité

résultat est encourageant parce qu'avoir un lien utilisé à plus de 80% en moyenne implique presque nécessairement un délai de file d'attente élevé et une perte probable de paquets. Ayant vu les effets directs du Coroutage sur les flots UDP, la prochaine étape consiste à montrer l'impact sur la performance des flots UDP en terme de délai et de gigue.

6.1 Analyse de qualité de service obtenue avec le Coroutage

Deux types de résultats vont être exhibés: les résultats globaux, et les résultats par Origine/Destination (O/D). Pour chaque type, le délai de file d'attente et la gigue sont enregistrés pour UDP, ainsi que le délai total et la gigue pour TCP. L'analyse globale des données procure une idée générale de l'amélioration de performance des flots UDP avec le Coroutage. Le tableau 6.2 illustre cette amélioration.

Pour chaque scénario, le $95^{ième}$ percentile et la valeur maximale sont indiqués. Ces valeurs sont comparées entre le Coroutage et le routage standard, pour le délai et la gigue. Par exemple, le $95^{ième}$ percentile de la gigue, pour le scénario A_3, est 6.59msec, lorsque le routage standard est utilisé. Ceci veut dire que 95% de toutes les valeurs de gigue sont inférieures à 6.59msec, pour le script A_3 et en routage standard. On compare cette valeur avec 0.36msec, qui est le $95^{ième}$ percentile des valeur de gigue pour A_3 en Coroutage, et on remarque que le Coroutage à amélioré la performance de 94%. En observant le reste du tableau, on peut remarquer que le Coroutage améliore grandement les $95^{ième}$ percentiles du délai et de la gigue, par rapport au routage standard. Ceci est le résultat de forcer autant que possible les flots UDP à aller sur des liens qui ne sont pas utilisés à plus de 50% en moyenne.

Le taux UL est une moyenne à long terme, ce qui veut dire que la bande passante va pour sûr fluctuer grandement à court terme autour de la valeur UL enregistrée

Scénario		Délai file d'att. 95ième %	Délai max. file d'attente	Gigue 95ième %	Gigue maximale
A_1	Standard	6.16ms	29.25ms	3.90ms	18.89ms
	Coroutage	0.52ms	18.11ms	0.79ms	16.14ms
	Amélioration	**91.6%**	**38.1%**	**79.7%**	**14.6%**
A_2	Standard	2.10ms	17.53ms	2.10ms	11.92ms
	Coroutage	0.25ms	18.18ms	0.40ms	12.67ms
	Amélioration	**88.1%**	**-3.7%**	**81.0%**	**-6.3%**
A_3	Standard	162.16ms	229.15ms	6.59ms	39.46ms
	Coroutage	0.25ms	163.02ms	0.36ms	30.84ms
	Amélioration	**99.9%**	**28.9%**	**94.6%**	**21.9%**
A_4	Standard	224.88ms	365.07ms	7.91ms	41.82ms
	Coroutage	0.15ms	318.23ms	0.21ms	42.37ms
	Amélioration	**99.9%**	**12.8%**	**97.4%**	**-1.3%**
A_5	Standard	75.43ms	140.95ms	5.94ms	31.20ms
	Coroutage	3.07ms	61.11ms	1.54ms	28.00ms
	Amélioration	**95.9%**	**56.6%**	**74.0%**	**10.3%**
A_6	Standard	192.87ms	279.05ms	6.35ms	36.42ms
	Coroutage	0.63ms	234.16ms	0.44ms	37.50ms
	Amélioration	**99.7%**	**16.1%**	**93.1%**	**-3.0%**
A_7	Standard	194.85ms	485.52ms	6.30ms	40.01ms
	Coroutage	3.87ms	434.27ms	1.10ms	42.47ms
	Amélioration	**98.0%**	**10.6%**	**82.5%**	**-6.1%**

Tableau 6.2 Amélioration de la qualité de service des flots UDP entre le routage standard et le Coroutage

et distribuée. Cependant, puisque le choix de liens convenables pour UDP et très conservateur à 50%, ceci implique que peu ou pas d'interfaces de sorties de ces routeurs ont experiencé un nombre entrant d'octets supérieur à leur capacité de ligne, et ce malgré les grandes fluctuations de bande passante. Ainsi, il y a eu très peu d'attente dans les files pour les paquets UDP. Pour ce qui est de la valeur maximale, les résultats varient. Que ce soit pour le délai de file d'attente ou pour la gigue, il y a des cas où la valeur maximale augmente avec le Coroutage. De plus, il n'y a pas de corrélation entre le délai et la gigue pour ces valeurs maximales. Par exemple, dans le scénario A_4 le délai est plus petit en Coroutage qu'en routage standard, alors qu'il est négatif pour la gigue maximale. D'un autre côté, entre les scénarios A_1 et A_2, ce qui a changé dans les scripts était seulement l'ensemble des nœuds utilisé pour les origines et destinations des flots, et pourtant les changements des délais et gigues maximaux est négatif pour A_2. Entre A_2 et A_3, une grande augmentation de trafic a eu lieu: les flots TCP sont passés de 5000 à 7000, et les UDP de 500 à 700. Pourtant, les valeurs maximales se sont améliorées en A_3 par rapport à A_2. La variabilité de ces résultats semble être due à la dynamique de la fenêtre de congestion du protocole TCP. Par exemple, une courte mais puissante rafale TCP peut causer un paquet UDP (présent dans le même routeur) d'expériencer un délai de file d'attente qui n'est pas ressenti par la plupart des autres paquets du même flot, ce qui fera en sorte que la valeur de gigue enregistrée ne sera pas représentative de la performance générale vue par les autres paquets.

Pour conclure, nous remarquons que seuls les deux derniers scénarios subissent une perte de paquets UDP, tel qu'illustré à la table 6.3. Vu l'amélioration du service offert aux UDP par le Coroutage, il n'est pas étonnant que les pertes de paquets soient éliminées losqu'elles existent. Dans A_6 et A_7 respectivement, le routage standard cause 0.032% et 0.26% de perte de paquets, et ces pertes n'existent plus

Scénario	Routage standard	Coroutage
A_1	0%	0%
A_2	0%	0%
A_3	0%	0%
A_4	0%	0%
A_5	0%	0%
A_6	0.032%	0%
A_7	0.26%	0%

Tableau 6.3 Pourcentage de perte de paquets UDP dans chaque scénario en Coroutage.

6.2 Analyse par Origine/Destination

Pour examiner plus en profondeur les effets du Coroutage, la performance des flots UDP est examinée au niveau Origine/Destination (O/D). Ici aussi, les mesures enregistrées sont le délai de file d'attente et la gigue. À la fin de chaque simulation (standard et Coroutage), la moyenne, le maximum et l'ecart type du délai de file d'attente et de la gigue sont disponibles par O/D. Ces valeurs représentent la performance des flots UDP utilisant cette paire O/D. Les résultats sont représentés sous forme d'histogramme, avec le temps en msec dans l'abscisse, et le nombre de paires O/D dans l'axe des ordonnées. Chaque histogramme montre soit la gigue, soit le délai de file d'attente par scénario. Les valeurs du routage standard et celles du Coroutage sont superposées dans un même histogramme.

Par exemple, dans la figure 6.1, le premier histogramme montre le délai de file d'attente moyen pour le scénario A_2, et le deuxième montre la gigue moyenne. On peut remarquer aisément que les barres reliées au Coroutage (les barres de couleur

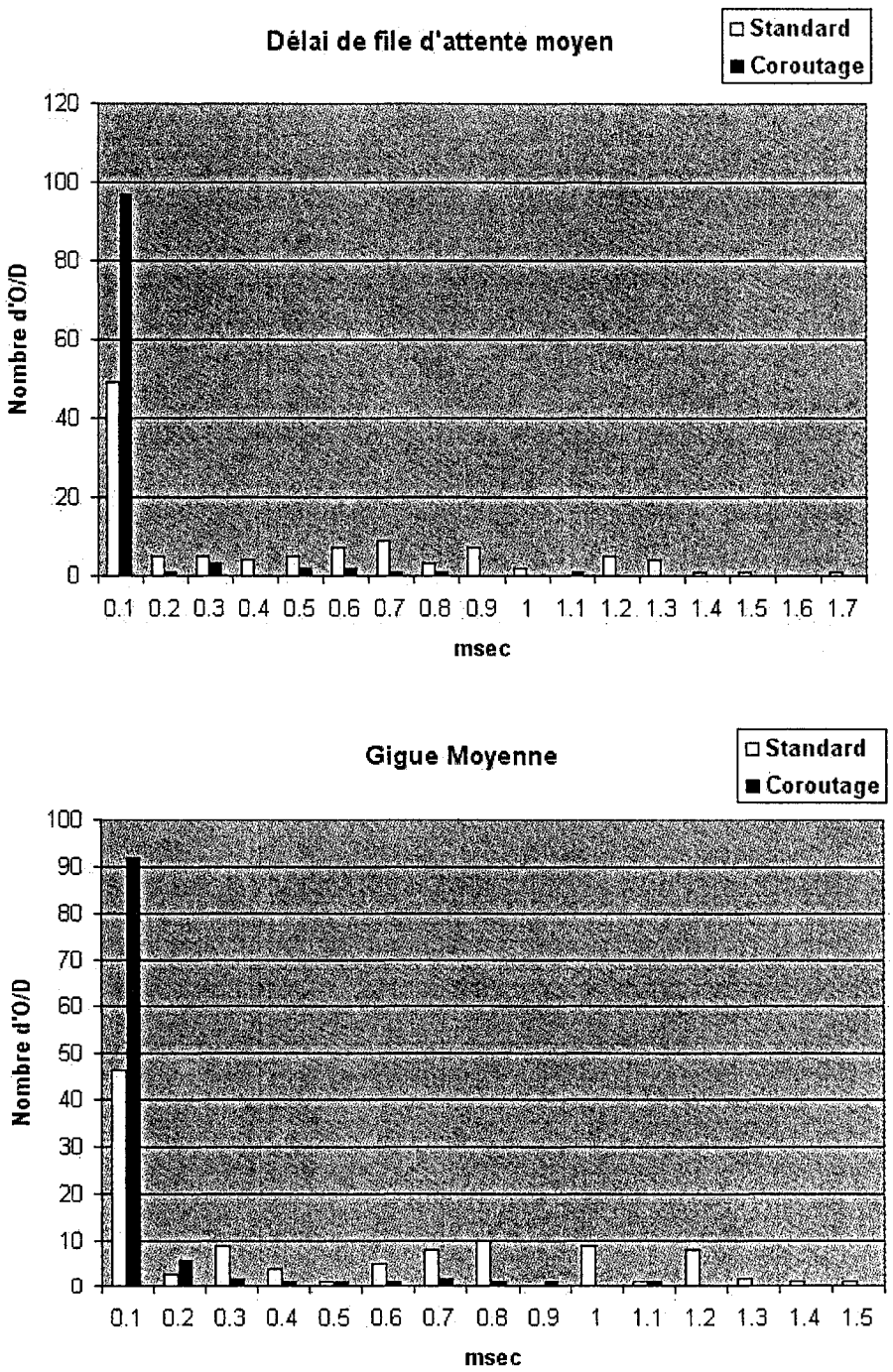


Figure 6.1 Histogrammes UDP du scénario A_2

sombre) sont plus concentrées vers la gauche. Dans l'histogramme de la gigue, plus de 90 des paires O/D totales ont une gigue moyenne de paquet d'au plus 0.1msec, comparé à 40 paires pour le routage standard. Tous les histogrammes pour les UDP, que ce soit pour la gigue ou le délai de file d'attente, montrent un résultat similaire, celui de valeurs dont la distribution est fortement condensée lors du Coroutage. Cette analyse corrobore les résultats globaux vus à la section précédente, et permet de conclure que le Coroutage affecte de façon positive la performance des flots UDP. A la fin du chapitre, nous incluons deux histogrammes supplémentaires à titre d'exemple.

6.3 Les intervalles de confiance lors de la comparaison de moyennes

Puisqu'on compare deux moyennes lors de l'analyse Origine/Destination, il est important de voir l'écart entre la moyenne et les valeurs elles-mêmes. En effet, si la distribution des valeurs d'une moyenne μ_1 chevauche celle d'une moyenne μ_2, alors comparer μ_1 et μ_2 est difficile. En conséquence, nous avons déterminé des intervalles de confiance à 95% pour les moyennes du délai et de la gigue de chaque O/D, pour les flots TCP et UDP, et pour tous les scénarios de simulation. La formule utilisée pour calculer un intervalle de confiance à 95% est la suivante:

$$I = \mu \pm \frac{1.96 \times \sigma}{\sqrt{n}} \tag{6.1}$$

Ainsi, si on on prend le délai des paquets UDP pour une certaine origine destination, les deux valeurs de I représentent les bornes supérieures et inférieures de l'intervalle de confiance des valeurs enregistrées de ce délai. μ représente la valeur moyenne obtenue, σ l'écart type, et n est le nombre de valeurs enregistrées durant la simulation. n variait selon les cas entre un minimum de plusieurs milliers

et jusqu'à plusieurs centaines de milliers, indiquant que les résultats obtenus sont valides.

Nous avons pu vérifier que la taille des intervalles est toujours petite par rapport à la valeur des moyennes. Ainsi, comparer la moyenne pour une O/D entre le Coroutage et le routage standard est statistiquement significatif. Une inspection visuelle a été faite pour tous les scénarios, et tous les cas avaient des intervalles suffisamment petits. La figure 6.2 en est un exemple. On y voit les valeurs de gigue moyenne (en secondes) pour chaque O/D. Les barres noires au-dessus et en-dessous de chaque point représentent les intervalles de confiance, indiquant que la comparaison de la moyenne du Coroutage avec celle du routage standard est valable.

6.4 Un cas particulier

Bien que l'effet que le Coroutage a sur les flots UDP est en majorité positif, ou du moins neutre, quelques cas sont apparus dans lequels le résultat moyen était pire dans le Coroutage que dans le routage standard. Voir par exemple la figure 6.3. La figure est un diagramme en barre, avec les O/D en abcisse, et la gigue moyenne en ordonnée. Pour simplifier le diagramme, nous avons seulement inclu les O/D dont la gigue moyenne en routage standard est d'au moins 3 msec. Les barres sombres indiquent les gigues avec Coroutage, et les barres claires indiquent les résultats avec routage standard.

Nous pouvons clairement voir qu'il y a deux paires O/D pour lesquelles la gigue moyenne est plus élevée d'à peu près 3.6% en Coroutage qu'en routage standard. Après examen, il s'avère que la raison est simplement que l'algorithme de routage explicite n'a pas pu trouver une route peu congestionnée pour les flots UDP de ces

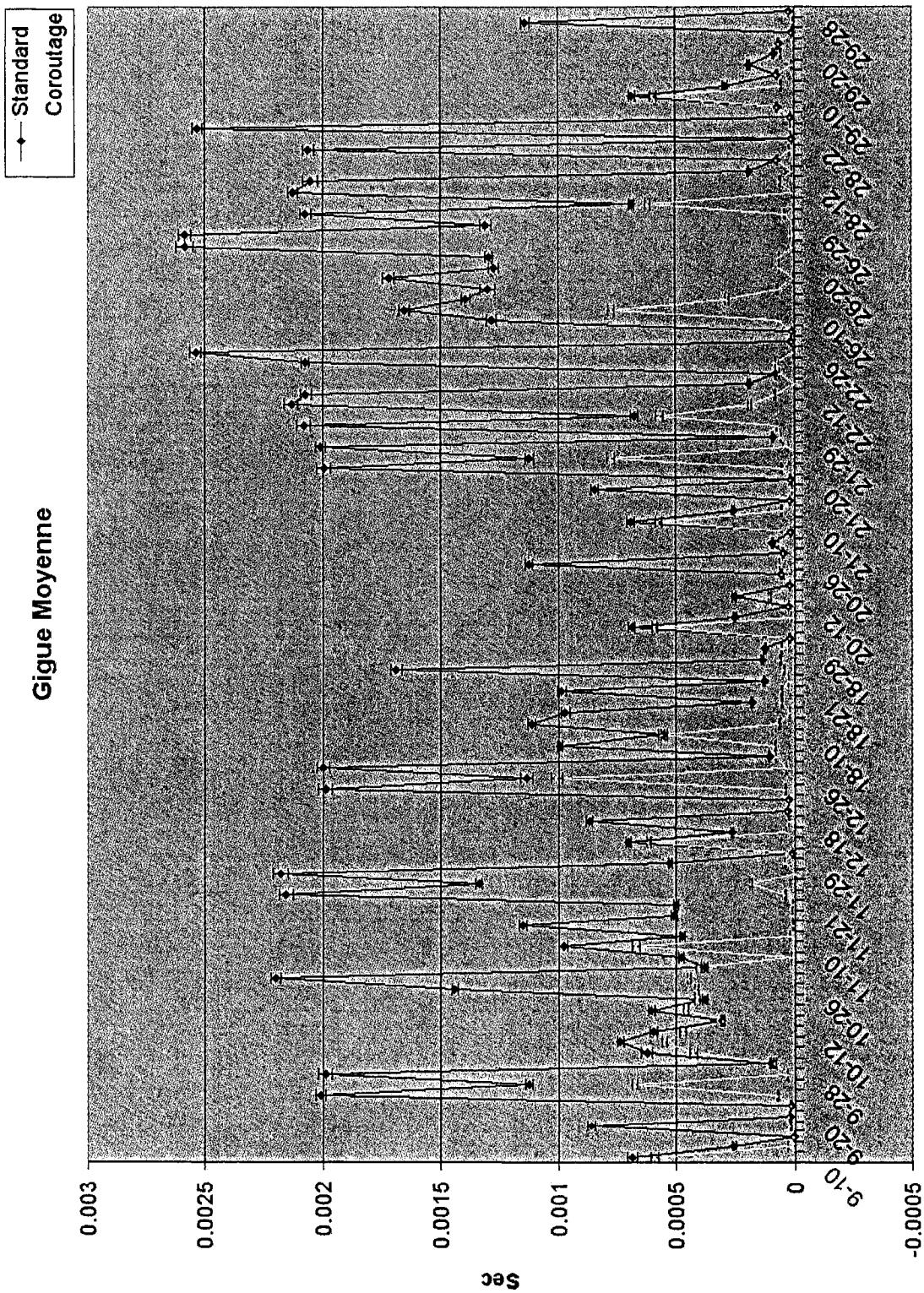


Figure 6.2 Exemple d'intervalles de confiance pour chaque O/D

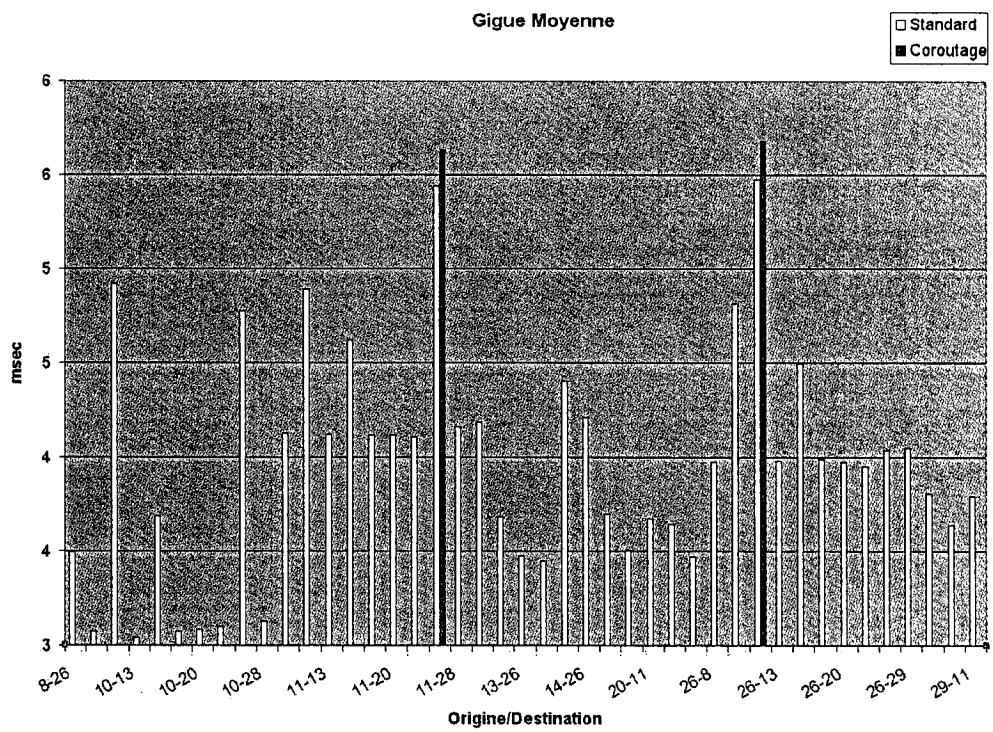


Figure 6.3 Gigue moyenne UDP par O/D (A_4)

paires O/D. Ainsi, ces flots ont suivi la route nominale de ces O/D, qui contiennent deux liens très congestionnés.

La raison pour laquelle la performance en Coroutage était pire, et non pas égale, à celle du routage standard est due aux flots TCP, et à leurs mécanismes de variation de taux d'envoi de paquet. En effet, quelques flots TCP sont capables d'envoyer plus de paquets en Coroutage qu'en routage standard, parce que les routes nominales sont libérées des flots UDP. Puisque les paquets TCP sont plus larges en octets que les paquets UDP, la conséquence est que plus d'octets iront sur les liens de la route nominale durant la simulation avec Coroutage. Ceci indique que le Coroutage peut modifier le comportement de TCP dans certains cas, et cette analyse sera examinée plus en profondeur à la section suivante. Il semble que dans ces rares cas, il y avait assez de différence en bande passante libérée sur les chemins nominaux pour rendre les flots TCP de ces chemins plus aggressifs.

Pour tester et voir si ces cas de performance réduite arrivent souvent, les scénarios A_5 à A_7 ont été conçus, avec plus de 50% de flots UDP. Bien que peu réalistes, ces scénarios permettent d'évaluer la fréquence des cas négatifs. Après simulation, le résultat était que seules deux paires d'origines/destinations, dans le scénario A_7, ont subi une performance négative, où la gigue moyenne était 5% et 13% supérieure en Coroutage qu'en routage standard. Puisque ces cas sont relativement rares même dans des scénarios extrêmes, on peut conclure que la probabilité d'en avoir est faible. Il faut en effet que plusieurs conditions aient lieu en même temps: ne pas pouvoir trouver une route explicite convenable pour un flot UDP donné; avoir des liens de la route nominale qui dépassent 100% de taux UL; avoir les flots TCP utilisant ces liens qui ouvrent leur fenêtres d'envoi plus qu'en routage standard; etc...

Pour conclure cette analyse des flots UDP en Coroutage, on peut dire que le résultat

est en majorité positif, et que le Coroutage est capable de réduire de façon significative la gigue et le délai pour les flots temps-réel. La prochaine étape consiste à voir comment le Coroutage affecte les flots TCP.

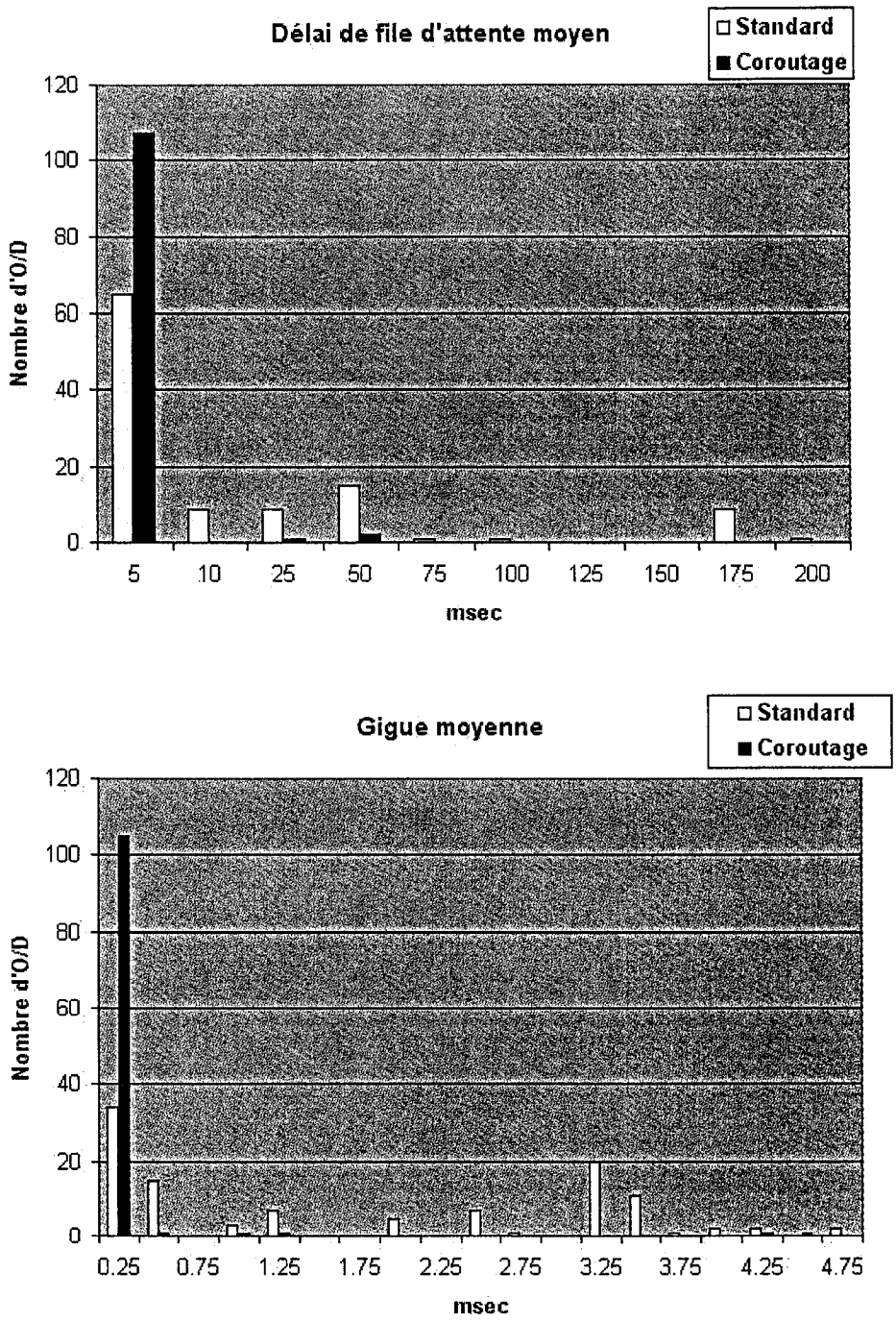


Figure 6.4 Histogrammes UDP du scénario A_3

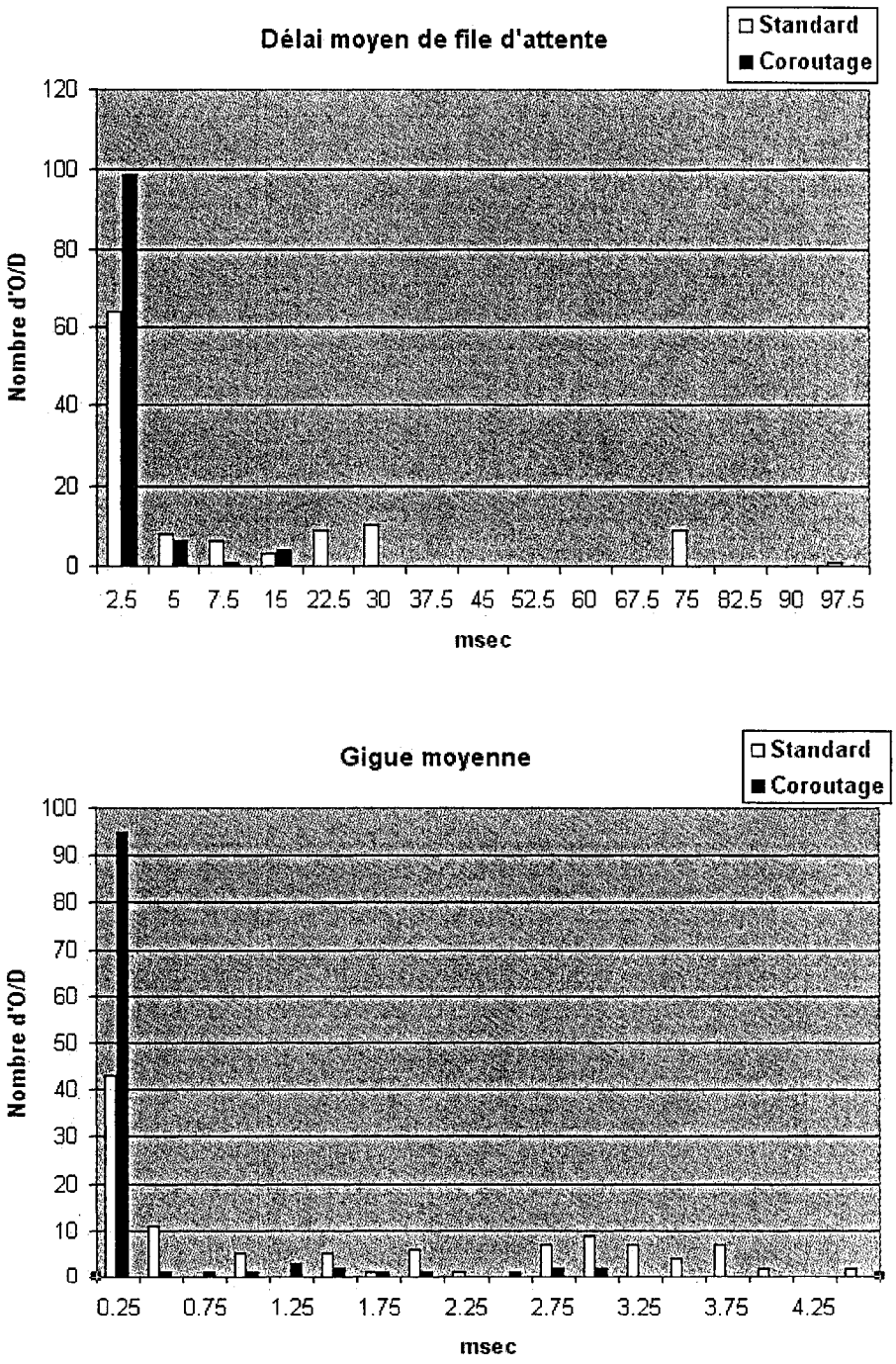


Figure 6.5 Histogrammes UDP du scénario A_5

CHAPITRE 7

ANALYSE TCP

Dans le Coroutage, les flots TCP suivent, comme en routage standard, la route nominale entre chaque paire O/D, et ceci parce que le Coroutage vise spécifiquement les flots UDP et tente d'améliorer leur performance. Cependant, il est important d'examiner l'effet de cette méthode de routage hybride sur la performance des flots TCP.

7.1 Résultats globaux

Pour avoir une première impression de l'effet du coroutage sur les TCP, le tableau 7.1 montre le $95^{ième}$ percentile et le maximum des valeurs de délai et de gigue. Puisque le chemin utilisé par un flot TCP ne change pas entre le routage standard et le Coroutage, il est possible de comparer le délai total (délai de file d'attente + transmission + propagation) dans les deux approches. Si l'on observe le $95^{ième}$ percentile du délai, on peut remarquer que l'effet du Coroutage est soit neutre (moins de 1% de changement), soit positif tel que dans le scénario A_6 (réduction de 5% de l'intervalle du $95^{ième}$ percentile). Ainsi les flots TCP semblent pouvoir s'adapter à des charges croissantes de traffic et/ou à des pourcentages croissants de flots UDP dans le réseau, ce qui fait que les files d'attentes ne se remplissent pas avec l'accroissement du trafic et les délais restent bas.

En ce qui concerne la gigue, le $95^{ième}$ percentile indique que les valeurs de gigue peuvent augmenter de façon assez significative même si les valeurs de délai décroissent,

Scénario		Délai total 95ième %	Délai max.	Gigue 95ième %	Gigue maximale
A_1	Standard	620.01ms	776.49ms	0.29ms	18.72ms
	Coroutage	620.01ms	774.12ms	0.28ms	16.93ms
	Amélioration	**0.0%**	**0.31%**	**3.5%**	**9.6%**
A_2	Standard	610.60ms	776.58ms	0.21ms	13.39ms
	Coroutage	610.57ms	781.67ms	0.20ms	13.40ms
	Amélioration	**0.0%**	**-0.7%**	**4.8%**	**-0.1%**
A_3	Standard	645.80ms	846.91ms	0.36ms	41.80ms
	Coroutage	632.44ms	836.31ms	0.37ms	28.13ms
	Amélioration	**2.1%**	**1.3%**	**-2.8%**	**32.7%**
A_4	Standard	671.28ms	897.28ms	0.41ms	64.71ms
	Coroutage	670.28ms	890.11ms	0.42ms	80.85ms
	Amélioration	**0.2%**	**0.8%**	**-2.4%**	**-24.9%**
A_5	Standard	620.38ms	814.48ms	0.34ms	41.64ms
	Coroutage	617.40ms	800.27ms	0.33ms	20.02ms
	Amélioration	**0.5%**	**1.7%**	**2.9%**	**51.9%**
A_6	Standard	672.53ms	968.84ms	0.37ms	130.38ms
	Coroutage	638.93ms	832.49ms	0.39ms	82.09ms
	Amélioration	**5.0%**	**14.1%**	**-5.4%**	**37.0%**
A_7	Standard	686.12ms	1025.33ms	0.37ms	128.52ms
	Coroutage	681.51ms	956.65ms	0.39ms	106.85ms
	Amélioration	**0.7%**	**6.7%**	**-5.4%**	**16.9%**

Tableau 7.1 Amélioration de la qualité de service des flots TCP entre le routage standard et le Coroutage

comme dans les scénarios A_6 et A_7. Dans A_6 par exemple, l'intervalle du $95^{ième}$ percentile des délais est réduit de 5% alors que celui de la gigue augmente de 5.4%. Nous remarquons aussi une plus grande variation pour la gigue maximale: en A_5, elle est réduite de 51.9%, alors qu'en A_4, elle augmente de 24.9%. Encore une fois, la dynamique des fenêtres de congestion des flots TCP serait responsable de ces variations. Cependant, il est important de noter que, d'après la colonne des 95%, les valeurs de gigue sont en majorité toujours inférieures à 1ms, quel que soit le scénario considéré. Nous pouvons ainsi conclure que le Coroutage n'est en général pas nuisible aux flots TCP. Puisque ce sont les flots UDP qui sont visés, il importait de vérifier que l'amélioration de la performance des flots UDP n'avait pas lieu au dépens des TCP.

7.2 Comptage des paquets

Si l'on observe maintenant la performance en terme de taux de perte de paquets TCP, les résultats montrent une amélioration quand le Coroutage est utilisé. En effet, le nombre total de paquets TCP qui traversent le réseau est compté pour chaque simulation, et est indiqué dans le tableau 7.2. Dans la colonne nommée *Nombre de paquets*, les changements positifs de pourcentage indiquent une augmentation du nombre de paquets ayant traversé le réseau. Dans la colonne *% perte de paquets*, le changement en pourcentage est positif lorsque le nombre de paquets perdus est réduit après Coroutage. Les résultats sont clairs: avec l'augmentation de la charge du trafic (de A_1 à A_4, et de A_5 à A_7), plus de paquets TCP ont pu être transmis par les sources de données. Ceci indique que le Coroutage a permis à quelques flots d'augmenter la taille de leurs fenêtres de congestion plus qu'il ne leur était possible sous routage standard, ce qui est en accord avec l'idée que les cas particuliers de mauvaise performance des flots UDP sont causés par la dynamique des

Scénario		Nombre de paquets	% perte de paquets
A_1	Standard	27,311,314	0.0
	Coroutage	27,320,360	0.0
	Amélioration	**0.0%**	**0.0%**
A_2	Standard	26,705,944	0.0
	Coroutage	26,708,022	0.0
	Amélioration	**0.0%**	**0.0%**
A_3	Standard	35,694,264	0.0
	Coroutage	35,991,018	0.0
	Amélioration	**0.8%**	**0.0%**
A_4	Standard	39,173,090	0.0
	Coroutage	39,343,322	0.0
	Amélioration	**0.4%**	**0.0%**
A_5	Standard	31,323,208	0.0
	Coroutage	31,756,768	0.0
	Amélioration	**1.4%**	**0.0%**
A_6	Standard	34,655,241	0.008
	Coroutage	35,836,302	0.000
	Amélioration	**3.4%**	**0.008%**
A_7	Standard	37,978,204	0.07
	Coroutage	39,791,490	0.00
	Amélioration	**4.8%**	**0.07%**

Tableau 7.2 Comptage des paquets TCP

flots TCP. Un autre élément remarquable est que le remplissage des files d'attente de sortie des routeurs est suffisamment réduit pour éliminer les pertes de paquets des scénarios A_6 et A_7.

En particulier, c'est surtout en A_5, A_6 et A_7 que le pourcentage d'augmentation de paquets est le plus significatif: il y a 0.4% plus de paquets dans le Coroutage en A_4, alors que le changement est de 1.38% en A_5. Ceci est clairement dû à l'augmentation du nombre de flots UDP de 400 en A_4 à 3000 en A_5. Puisque le Coroutage achemine des flots UDP qui iraient normalement sur des routes nominales congestionnées, ces dernières verront une réduction de l'utilisation de leur bande passante, ce qui permettra aux flots TCP d'augmenter leur fenêtre. Il est important de rappeler que les trois derniers scénarios ne sont pas réalistes, à cause du nombre élevé de flots UDP. Cependant, tous les scénarios montrent que la performance des flots TCP n'est pas pénalisée en Coroutage, et qu'au contraire il y a toujours un peu plus de paquets qui peuvent être transmis par rapport aux simulations sous routage standard.

7.3 Analyse par O/D

Les résultats par O/D illustrent les délais et la gigue moyens pour chaque paire Origine/Destination. Les histogrammes des figures 7.1, 7.2, et 7.3 sont semblables à ceux de l'analyse UDP. Ces figures montrent à nouveau que l'effet général du Coroutage sur les flots TCP n'est pas très significatif, avec de légères variations selon l'O/D. Ainsi, Les résultats des analyses UDP et TCP nous révèlent que les mécanismes du Coroutage sont capables d'offrir une amélioration notable aux flots prioritaires, tout en maintenant en général la performance des autres flots telle qu'elle l'était en routage standard.

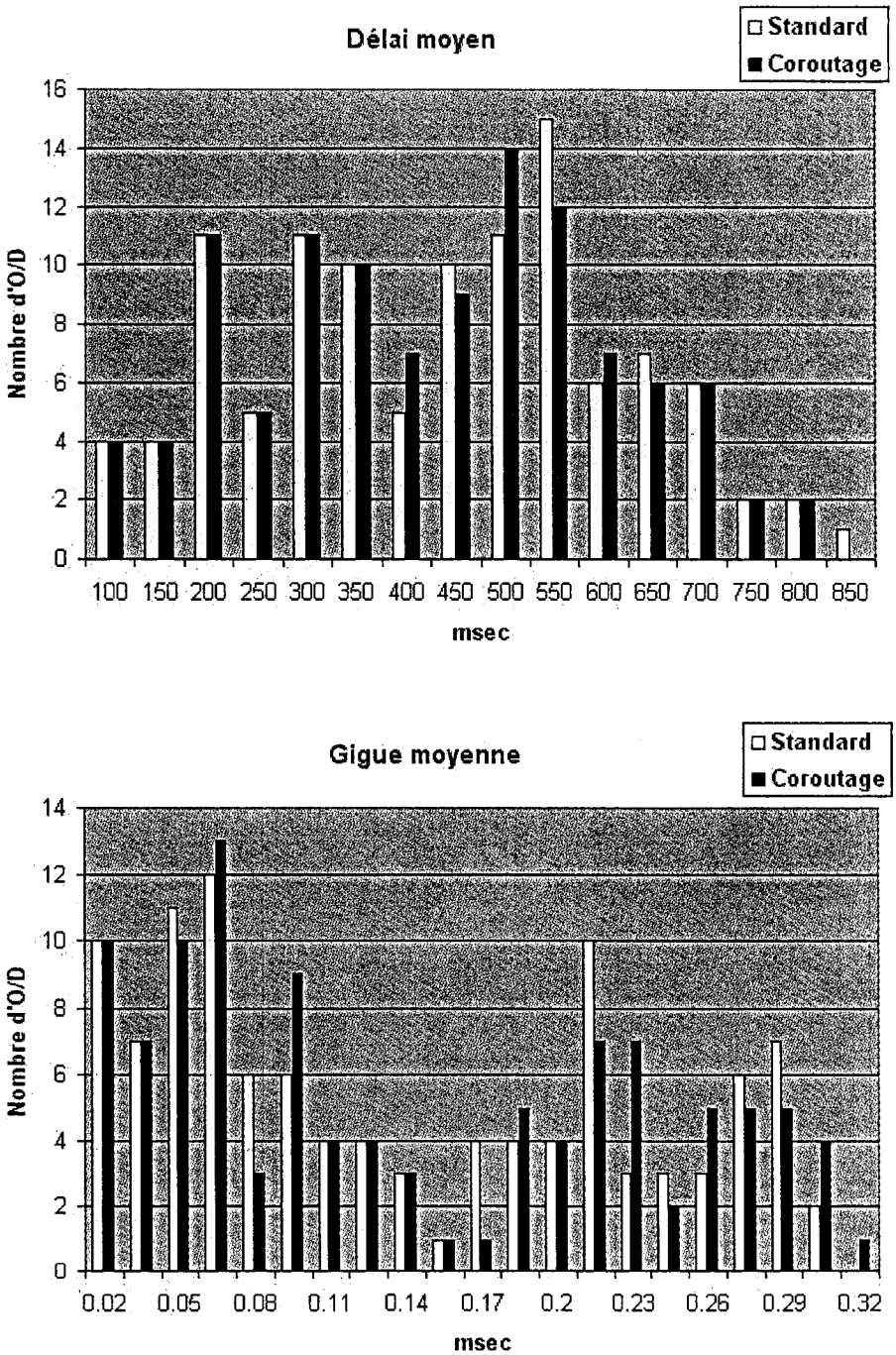


Figure 7.1 Histogrammes TCP du scénario A_3

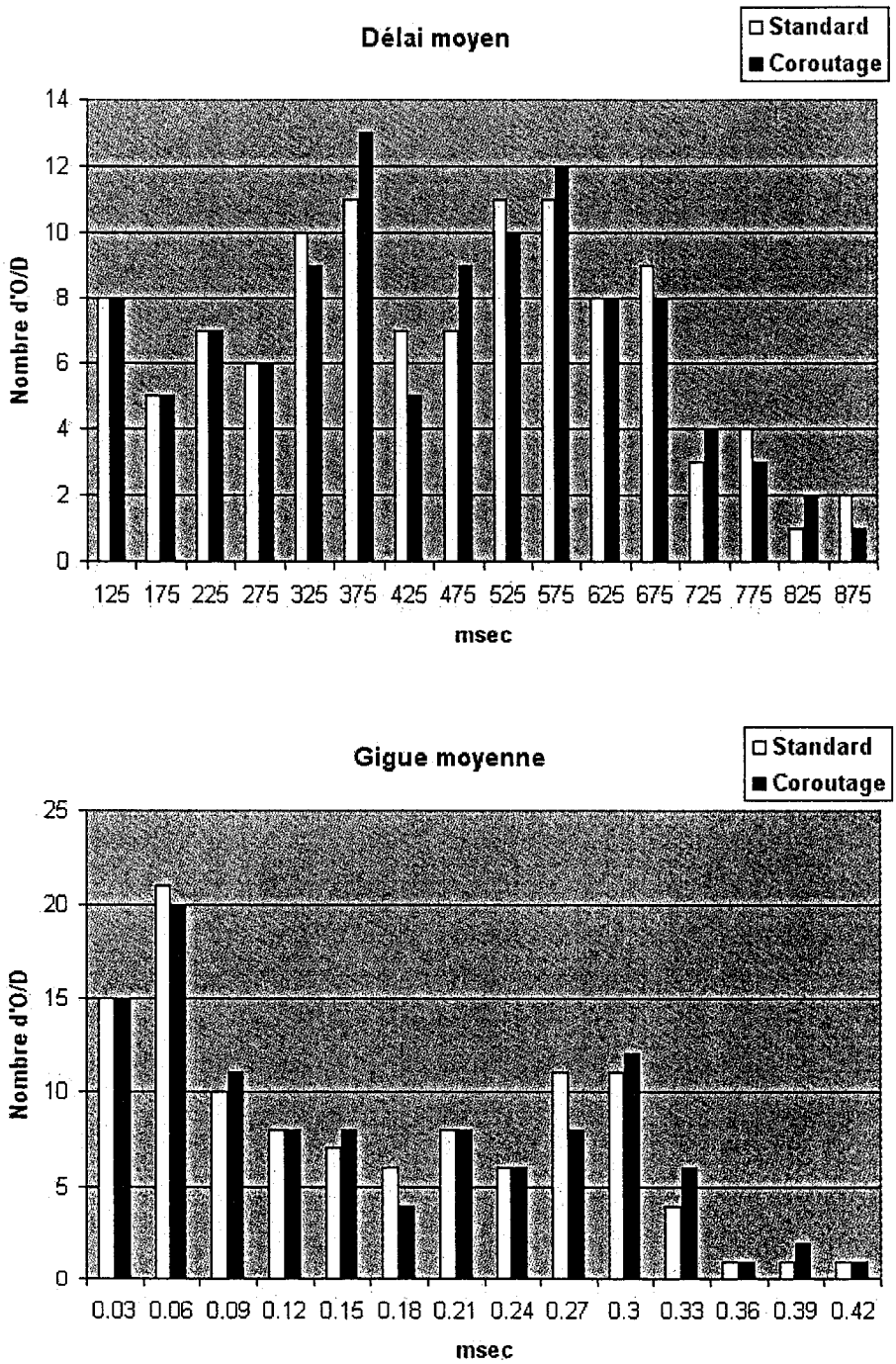


Figure 7.2 Histogrammes TCP du scénario A_4

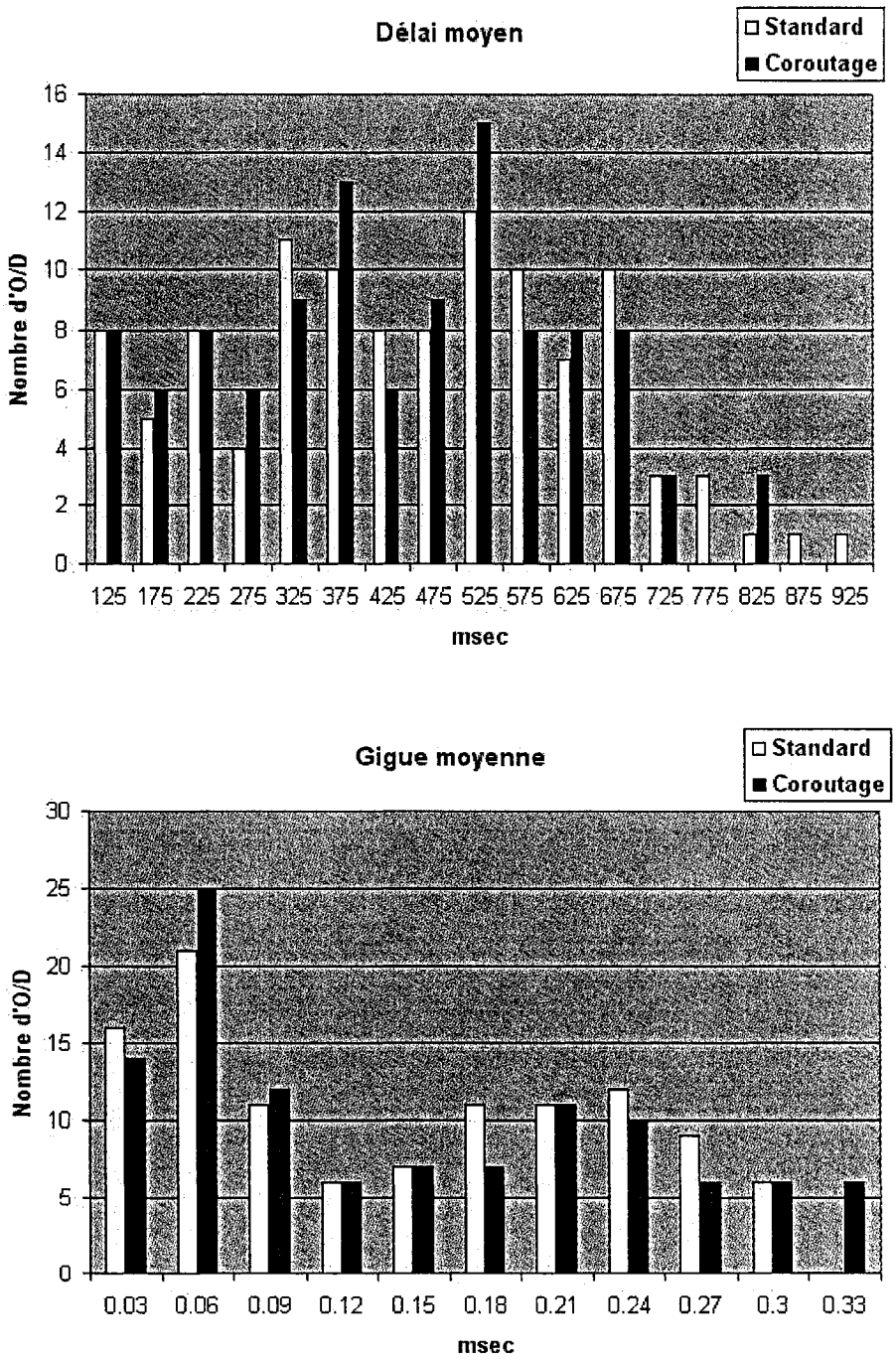


Figure 7.3 Histogrammes TCP du scénario A_6

CHAPITRE 8

ANALYSE DE DEUX TYPES D'ENCODAGE EXPLICITE

Pour pouvoir réaliser le Coroutage, il faut encoder la route explicite. Dans cette section, nous présentons deux méthodes d'encodage. Les deux méthodes utilisent des labels, que nous définissons plus en détail dans la section suivante. Nous n'avons pas décidé laquelle des deux méthodes d'encodage est la plus appropriée, laissant cette décision à une recherche ultérieure.

8.1 Précisions sur le terme *label*

Le terme label a été répandu avec l'apparition du standard MPLS [Rosen01]. Une caractéristique fondamentale de la notion de label est qu'elle permet de commuter un paquet rapidement lors de son passage dans un routeur. Pour mieux comprendre cet aspect, il importe de se rappeler comment la commutation se fait dans un routeur implémentant un routage IP classique. Les routeurs actuels contiennent des tables de routage d'environ 100,000 éléments. Chaque élément contient une adresse IP avec un masque de sous-réseau. A chacun de ces éléments correspond un indicateur d'interface de sortie du routeur. En gros, lorsqu'un paquet IP entre dans le routeur, ce dernier examine l'adresse IP de destination encodée dans l'en-tête du paquet. Le routeur compare ensuite cette valeur avec les adresses IP dans sa table. L'adresse qui est la plus proche de l'adresse dans le paquet est choisie, et le paquet est transmis via le port de sortie correspondant du routeur. Ceci est appelé le LMS (Longest Match Search). Le problème de performance est immédiatement apparent car il faut comparer beaucoup de valeurs dans la table avant de trouver

la valeur la plus proche.

MPLS, comme le Tag Switching de Cisco, se proposaient d'accélérer la commutation des paquets dans les routeurs en ajoutant un label (une séquence de bits) entre les en-têtes de niveau 2 et 3 de chaque paquet IP. Ces labels sont de taille fixe et correspondent chacun à un élément *unique* dans la table de routage de chaque routeur.

Dans ce texte, nous avons défini deux possibilités d'encodage de route explicite qui utilisent chacun une structure de label différente. Nous devrons éventuellement choisir entre l'une des deux méthodes. Quel que soit le choix, nous bénéficierons d'une commutation plus rapide pour les paquets routés explicitement que pour les paquets routés avec IP classique utilisant le Longest Match Search. Par la suite, nous allons nous référer à la première méthode d'encodage de labels comme étant la méthode d'encodage par Séquence d'Interfaces de Sortie (encodage SIS) et à la deuxième méthode comme étant la méthode des XOR.

Dans la prochaine section nous introduisons la méthode SIS avec ses avantages et ses inconvénients. Cette méthode contient des similarités avec la structure de labels MPLS, ce qui nous a poussé à comparer les deux structures. Nous expliquerons ensuite nos raisons pour examiner une deuxième méthode, celle des XOR, que nous expliquerons à la section 8.3.

8.2 Encodage avec la méthode SIS

Nous rappelons qu'un chemin explicite suivi par un flot n'est qu'une suite de routeurs. Une définition équivalente est de considérer tout chemin comme étant une séquence d'interfaces de sortie de routeurs. Comme chaque routeur doit transmettre un paquet vers un voisin, cette transmission est faite via un port, ou interface

de sortie. Ainsi, si un paquet rentre par un routeur A, traverse les routeurs X, Y et Z, avant de sortir du réseau via le routeur B, et si le paquet est transmis à travers le port de sortie p1 de X, le port p2 de Y, le port p3 de Z, et le port p4 de B, alors le chemin explicite est entièrement défini par la séquence p1-p2-p3-p4 à partir du routeur A. Tout paquet entrant par A se voit encoder la séquence du chemin qu'il doit suivre. Le routeur A détermine que le prochain saut du paquet est X et encode la route explicite (A est un routeur d'accès, et donc *edge stateful*, il peut mémoriser les états des flots). Le paquet est transmis vers X. Ensuite, X lit la première valeur dans l'encodage (qui est p1), et détermine donc par où il doit transmettre le paquet. Le routeur X élimine ensuite la valeur p1 de l'encodage. En conséquence, lorsque le paquet arrive vers Y, la première valeur de l'encodage est p2, et Y peut alors transmettre le paquet. Cette opération se répète tout au long de la traversée du paquet dans le réseau. Chaque label contiendra un seul numéro de port, et l'encodage sera donc formé, pour chaque paquet, d'une séquence de labels, chacun pointant vers un numéro de port, d'où le terme encodage SIS.

La taille de l'encodage de la route explicite dépend de la longueur de la route, puisqu'il y a une correspondance directe entre le nombre de routeurs du chemin et le nombre de labels à ajouter dans chaque paquet. Cette question de "overhead" doit être prise en considération, et c'est pourquoi nous allons introduire dans la section suivante le label le plus connu, celui du standard MPLS, pour illustrer le problème du "overhead". Nous nous en sommes inspirés dans la conception de notre propre label, que nous appellons label SIS, et que nous allons aussi voir en détail.

8.2.1 Le label du standard MPLS

L'IETF fixe à 20 bits la taille d'un label MPLS. Chaque label forme, avec 8 bits de TTL (Time To Live, ou nombre de sauts avant rejet), 3 bits EXP, et 1 bit *bottom of stack* (indicateur du fond de la pile), le shim header (voir figure 8.1). Ainsi, insérer un label implique insérer un shim header, avec la possibilité d'en empiler plusieurs par paquet. Supposons que nous voulions utiliser des label MPLS dans l'encodage de la route explicite, avec un label par port de sortie de routeur. Nous serions obligés d'encoder dans chaque paquet une séquence de shim headers de 4 octets chacun. A titre d'indication, sachant que le "overhead" des en-têtes TCP et IP est d'environ 40 octets par paquet, et si nous avons une route explicite de 10 routeurs, nous aurons 40 octets pour la route explicite à ajouter au paquet, et avons donc multiplié par 2 le "overhead". En bref, le shim header MPLS est trop grand pour nos besoins, d'où la nécessité d'introduire notre nouveau label SIS.

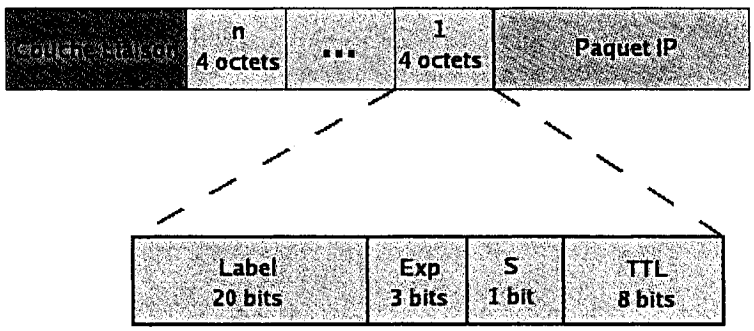


Figure 8.1 Shim Header standard de MPLS

8.2.2 Encodage avec labels SIS

La grande taille du shim header de MPLS nous a inspiré notre concept de label SIS: tout paquet routé explicitement contiendra dans son en-tête, entre la partie liaison et la partie IP, une pile de labels SIS. Chaque label SIS aura une taille de

1 octet. Le premier label contient une valeur de TTL, pour éviter les cycles, et chacun des autres labels contiendra le numéro d'une interface de sortie de routeur. La figure 8.2 illustre notre concept. Les 8 bits par label de port de sortie permettent de définir 256 interfaces de sortie possibles par routeur. S'il s'avère que les routeurs auront besoin de plus d'interfaces, les labels pourront être agrandis. Dans le cas contraire, enlever des bits pour chaque label diminuera le "overhead". A titre de comparaison avec la section précédente, si nous avons une route explicite formée de 10 routeurs, avec un octet par routeur, nous avons dans ce cas 11 octets qui formeront le "overhead" de l'encodage SIS de la route explicite, soit 10 octets pour 10 routeurs plus un octet pour le TTL, par opposition à 40 octets avec le shim header MPLS auparavant.

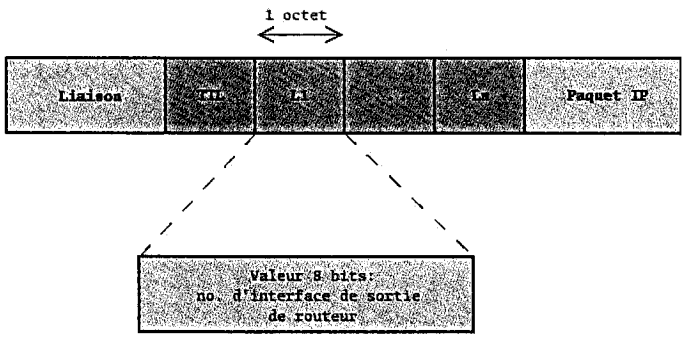


Figure 8.2 Encodage avec labels SIS

Il faut aussi étudier la complexité de cet encodage. Lorsqu'un paquet arrive à un routeur avec un encodage SIS, il appartient à un flot déjà établi et a donc terminé avec succès la phase d'initialisation du routage explicite. Ce paquet aura besoin alors de très peu de temps de traitement dans le routeur. En effet, le traitement par paquet contenant un tel encodage se résume à de simples opérations:

```
pop, read, push, forward
```

Plus précisément, le routeur extrait le label du TTL, puis extrait le label suivant.

Ce dernier contient le numéro de port par lequel ce routeur doit transmettre le paquet en question. Le routeur n'a plus qu'à lire la valeur de ce label, de réduire de 1 le TTL puis de l'insérer, et enfin de transmettre le paquet dans la bonne direction. Pour mieux comprendre ces mécanismes, se référer à la figure 8.3.

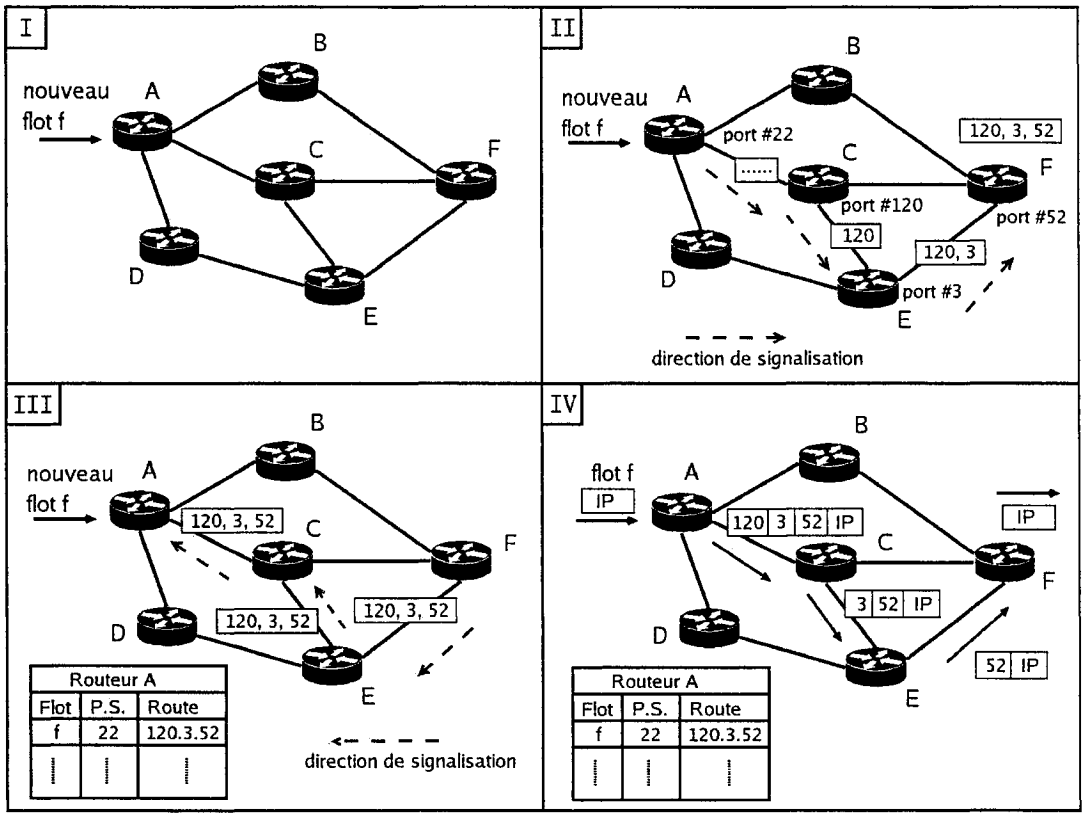


Figure 8.3 Routage explicite avec SIS

Dans cette figure, les trois premières parties illustrent la phase d'initialisation du nouveau flot f, où le sous-protocole de routage explicite détermine que le flot doit suivre la route ACEF, et doit donc encoder dans les paquets de f la séquence de labels 120-3-52. Ensuite, les paquets de f commencent à être transmis, tel qu'indiqué dans la quatrième partie de la figure. Les routeurs sur le chemin explicite de f utilisent les informations présentes dans l'encodage SIS des paquets, en lisant

chacun le numéro de port dans le label qui lui est destiné. Nous voyons clairement dans la figure que le routeur C extrait le label avec valeur = 120, et en déduit qu'il doit transmettre le paquet à travers son port 120. Il en va de même pour les routeurs E et F, qui transmettent respectivement le paquet à travers les interfaces 3 et 52. Par souci de clarté, le label TTL n'a pas été illustré.

Une faiblesse de l'encodage SIS est que le "overhead" de la route explicite est proportionnel à la longueur de la route, et donc variable, avec le risque d'avoir une grande taille d'en-tête. Un avantage est que les routeurs du cœur du réseau n'ont à mémoriser aucune information sur l'état des flots routés explicitement. Ceci répond à notre exigence d'avoir un routage *core stateless.*

Une autre faiblesse de SIS est que cet encodage propose une nouvelle structure de label, ce qui céerait de la résistance à son adoption. Si le problème du "overhead" et de la nouvelle structure de label s'avère être important, on peut utiliser un autre encodage, que nous nommons encodage XOR, et qui permet d'encoder la route explicite sur *un seul label* seulement. Nous allons voir qu'il faudra faire un compromis entre la réduction de taille de "overhead" et l'augmentation de la complexité de traitement et de la mémoire requise à chaque routeur.

8.3 Encodage par XORs successifs

Cette approche est tirée de la thèse de doctorat d'Ion Stoica, [Stoica00] section 3.1.3.3. Le tableau 8.1 offre un rappel de l'opérateur XOR, qui agit bit par bit sur deux nombres binaires.

Stoica propose l'idée suivante: chaque routeur est identifié par une séquence unique de bits. Quant à la route explicite pour un certain flot, elle est aussi identifiée par une séquence de bits, qui est créée en effectuant des XOR successifs sur les

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

Tableau 8.1 L'opérateur binaire XOR

identificateurs des routeurs qui forment le chemin du flot.

Ainsi, si un paquet traverse la route composée des routeurs R_a, R_b... jusqu'à R_n, alors le chemin de ce paquet aura la séquence binaire suivante ($\otimes$ représentant l'opération XOR) :

$l = R_a \otimes R_b \otimes R_c \otimes \ldots \otimes R_n$

Tous les paquets du flot traversant cette route contiendront la séquence l. À chaque routeur du chemin, un XOR est effectué entre cette séquence et l'identificateur du routeur, utilisant la propriété suivante:

Si $l = R_a \otimes R_b$ alors $l \otimes R_a = R_b$

Ainsi, refaire un XOR entre l et une valeur incluse dans l élimine cette valeur. Chaque routeur R_x sur la route définie par le label l garde la séquence $l \otimes R_x$ dans une table, et fait une correspondance de cette valeur à l'interface de sortie pour cette route.

La figure 8.4, tirée de la thèse de Stoica, illustre la méthode. Les routeurs sont identifiés par 4 bits. Les flots 1 et 2 suivent la même séquence de routeurs si on exclue les routeurs d'entrée (qui n'entrent pas dans l'élaboration du label). Les

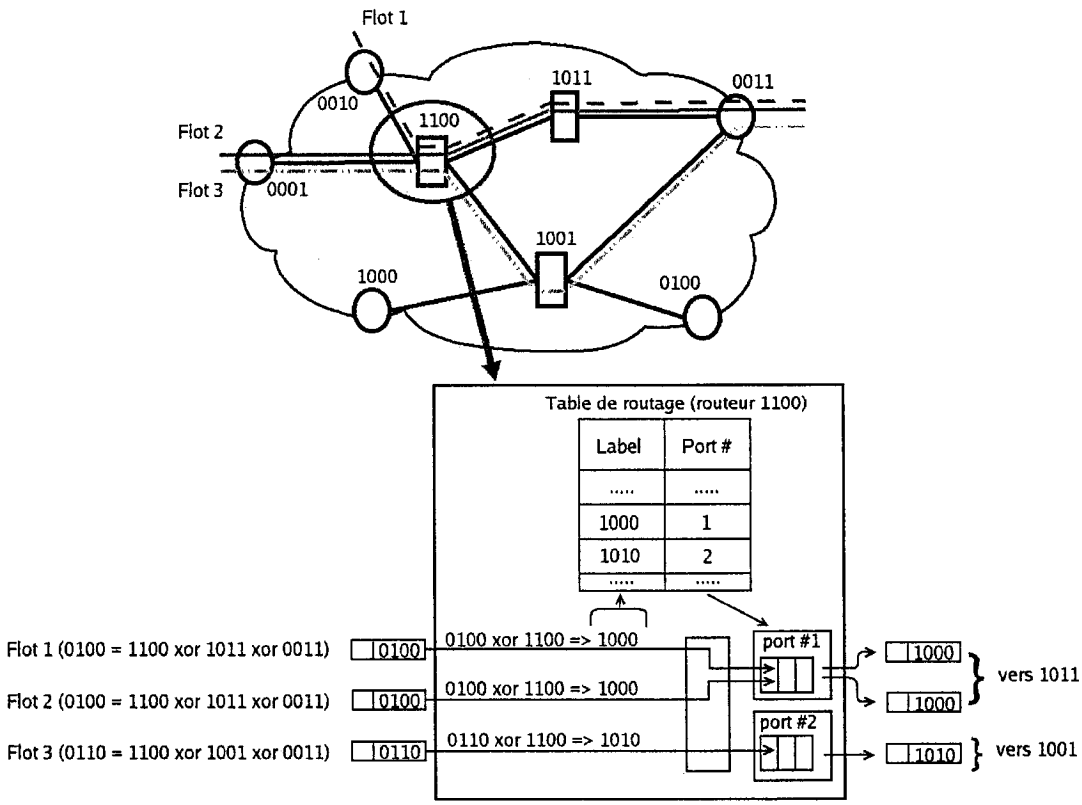


Figure 8.4 Exemple sur l'encodage par XOR successifs

paquets de ces flots contiennent donc le même label ($l = 0100 = 1100 \otimes 1011 \otimes 0011$). Le flot 3, quant à lui, suit un autre chemin ($l = 0110$).

Le routeur décrit dans la figure révèle les mécanismes qui ont lieu: les labels 0100 et 0110 sont XORés avec l'identificateur de ce routeur (1100). Le résultat obtenu est recherché dans une table qui fait la correspondance entre les valeurs binaires et les interfaces de sortie. Dans cet exemple, les flots 1 et 2 suivent le même chemin: ils auront tous deux la valeur 1000 qui correspond au port 1, alors que le paquet du flot 3 sortira du port 2 avec le label 1010.

Implicitement, cette méthode opère de façon similaire à celle de l'élimination successive de labels de l'encodage SIS: tout paquet entrant dans le réseau commence

avec un label formé du XOR de tous les routeurs sur son chemin. Puis, au fur et à mesure que le paquet avance dans le réseau, son label est à chaque fois 'réduit' par un XOR pour représenter le XOR des routeurs non encore traversés.

L'avantage de cette méthode est que le "overhead" dans chaque paquet *est de taille constante.* Par exemple on peut utiliser un seul label MPLS standard de 20 bits et le "overhead" serait donc un seul shim header de 4 octets, pouvant encoder 2^{20} routes. Ce label serait suffisant pour notre encodage XOR. Nous rappelons que la seule raison pour laquelle nous reprenons le label de MPLS est que certaines compagnies de télécoms pourraient être plus réceptives à des structures de label qui existent déjà.

Cette idée du XOR des identificateurs des routeurs a été décrite dans la thèse d'Ion Stoica. À partir de maintenant, les observations et propositions que nous allons mentionner sont les nôtres et, ayant examiné de plus près l'encodage XOR, nous voyons qu'il présente au moins deux inconvénients.

D'abord, la complexité par rapport à l'encodage SIS augmente du fait de la nécessité d'implémenter une table de correspondance. Ceci ne veut pas dire que nous sommes revenus au paradigme du *core stateful*, dans lequel chaque routeur traite les informations pour chaque état de flot, situation que nous cherchons justement à éviter. En effet, cette table aura une taille proportionnelle *au nombre de chemins explicites dans le réseau*, qui traversent le routeur en question, et non pas *au nombre de flots* qui empruntent le même chemin. Comme nous l'avons vu dans l'exemple précédent, les flots 1 et 2 qui empruntent le même chemin avaient tous deux le même label à leur arrivée au routeur concerné, et donc un seul élément dans la table était nécessaire. Ainsi un grand nombre de flots seront routés sur un nombre relativement petit de chemins. Par ailleurs, le temps de calcul reste faible puisque les opérations XOR sont très simples, et on n'accède à la table qu'une fois par

paquet.

Le deuxième inconvénient est plus délicat: il est dû au fait que l'opérateur XOR est commutatif, et donc par exemple la séquence: "$R_a \otimes R_b \otimes R_c$", et la séquence "$R_a \otimes R_c \otimes R_b$" donneront la même valeur de label. Ainsi, toutes les routes explicites formées de combinaisons différentes d'un *même ensemble de routeurs* auront *le même label* encodé dans les paquets des flots traversant ces chemins. Stoica considère que la probabilité d'avoir ces cas de collision n'est pas importante. Cependant, nous illustrons à la figure 8.5 une telle collision par l'exemple suivant.

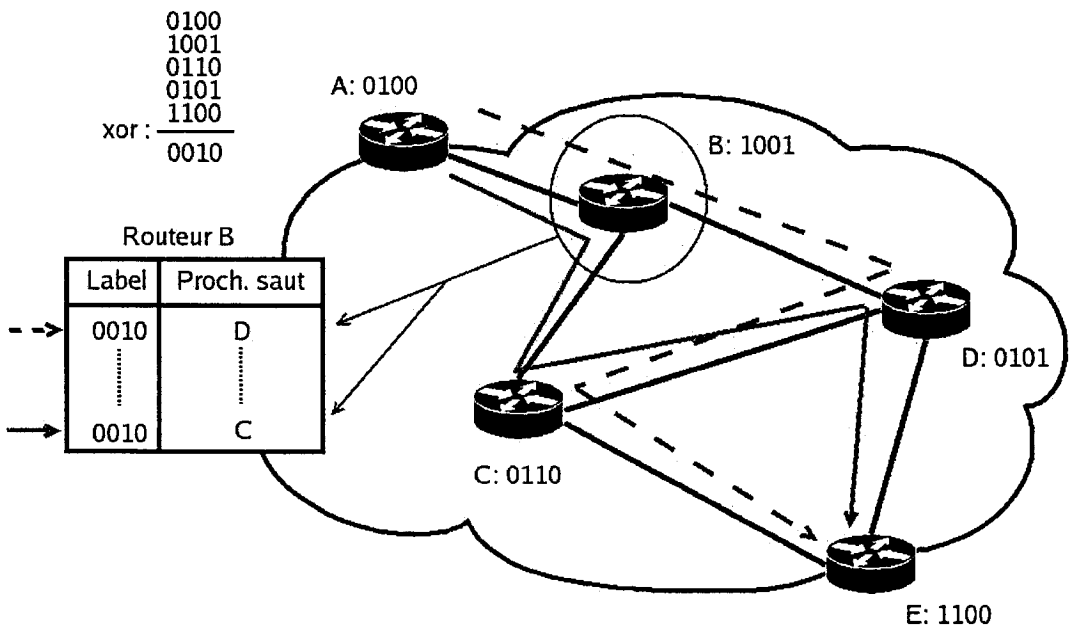


Figure 8.5 Exemple de collision de labels avec la méthode de Stoica

Les deux flots traversent le même ensemble de routeurs (A, B C, D et E), mais dans un ordre différent, ABDCE pour l'un et ABCDE pour l'autre. Par la méthode de Stoica, et si la table des labels du routeur B accepte les éléments identiques, les deux chemins seront décrits par le même label, 0010 dans l'exemple, dans deux

entrées différentes de la table. Les paquets des deux flots arriveront donc au routeur B avec un label identique, et il y aura collision. Le routeur B ne peut pas savoir quel paquet appartient à quel flot. La réaction en pratique de B dépendra de l'implémentation de l'algorithme dans le routeur.

Stoica considère que ce type de collision serait rare, sans offrir plus d'explications, en ajoutant aussi que si une collision a lieu, le routeur n'a qu'à choisir la première entrée dans la table pour tous les paquets. Ceci voudrait dire dans notre exemple que les deux flots seront routés sur un même chemin. Stoica conclut que, pour un routage par flot et donc d'une bande passante relativement petite, sur une route différente que celle décidée par l'algorithme de routage, il n'y aura pas d'influence majeure sur le réseau. De notre côté, nous considérons qu'il ne faut pas sous-estimer cette question avant d'avoir fait des tests plus poussés, et d'avoir établi si les collisions sont un phénomène fréquent ou non.

Une solution pour réduire les collisions de l'encodage XOR serait de réduire la probabilité de collision en faisant aussi un XOR successif des ports de sortie des routeurs, et ce en notant les numéros de ports de sortie des routeurs formant le chemin explicite, en plus des identificateurs des routeurs. Ensuite, au routeur d'accès du flot en question, faire le XOR de ces ports entre eux.

Le résultat est ainsi encodé dans chaque paquet du flot, à côté du résultat du XOR des identificateurs des routeurs. Si l'on prend un label de 20 bits, 8 pour les numéros de ports de sortie, et 12 pour les identificateurs des routeurs, ce label sera composé, pour un flot entrant dans le réseau, du XOR de tous les ports de sortie du chemin explicite *en plus* du XOR des identificateurs des routeurs. Arrivé à un routeur, un paquet de ce flot aura seulement la partie de son label contenant le XOR des identificateurs des routeurs qui sera XORé avec l'identificateur local du routeur en question. On ne peut pas XORer la partie des ports de sortie, puisqu'on

ne sait pas avec quel numéro de port le faire. La partie des ports de sortie reste inchangée durant la traversée du paquet dans le réseau.

Avec cette variation de la méhode des XOR, la table de correspondance des routeurs contiendrait des entrées plus détaillées par chemin explicite, ce qui devrait réduire la probabilité de collision. En effet, lorsque deux flots traversent le même ensemble de routeurs, il est moins probable qu'ils traversent aussi le même ensemble de ports, et auront donc deux entrées différentes dans les tables des routeurs. Le risque de collision est réduit puisqu'on a différencié ainsi les chemins explicites de deux façons au lieu d'une, mais la probabilité en est toujours *non nulle.*

L'encodage par XOR est donc intéressant du point de vue de la réduction de l'"overhead" d'encodage de la route explicite dans chaque paquet. Cependant, plusieurs questions demandent plus d'analyse afin de vérifier la robustesse d'un tel encodage. En conclusion, le tableau 8.2 résume et compare les méthodes d'encodage SIS et XOR.

	Méthode SIS	Méthode des XOR
"Overhead"	Variable	Constant
Commutation	Extraire le TTL Décrémenter le TTL Extraire le label Lire le label Insérer le TTL Transmettre le paquet	Xor du label Lire la table Transmettre le paquet
Structure de label	Nouvelle structure nécessaire	Possible d'utiliser le shim header MPLS
Addition de mémoire dans les routeurs	Aucune addition nécessaire	Addition d'une table de labels
Autre	———	Risque de collision de labels

Tableau 8.2 Résumé des deux méthodes d'encodage proposées

CONCLUSION

Le Coroutage est un protocole de routage qui route explicitement les flots UDP temps-réel, et classiquement les flots TCP. Le Coroutage est ainsi une méthode hybride, combinant deux approches de routage. Ayant analysé d'abord les caractéristiques des ces deux types de protocoles de transport, nous avons conclu après simulation que le Coroutage peut améliorer sensiblement la performance des flots UDP en terme de délais et de gigue. De plus, cette amélioration ne s'est pas faite au dépens des flots TCP, pour lesquels l'effet du Coroutage est relativement neutre. L'aspect novateur de notre recherche est de décider de séparer les flots TCP et UDP, basé sur leurs différences en tant que protocoles de transports.

Le Coroutage peut ainsi augmenter efficacement la robustesse et l'extensibilité dans les réseaux de communication. En effet, en visant les flots UDP temps-réel et en les routant explicitement loin des liens sur-utilisés, nous avons pu réduire un peu la pression sur ces liens congestionnés. Sachant que beaucoup de flots UDP sont assez inflexibles et ne réduisent pas leurs taux d'envoi lorsqu'un lien devient surchargé, nous pouvons ainsi protéger les liens qui ont tendance à être surchargés. Vu la croissance des flots UDP due à la popularité des applications de Voix et de Vidéo sur IP, cet aspect du Coroutage garantit une certaine robustesse au réseau.

Par rapport à d'autres approches qui considèrent le déséquilibre de distribution de la charge du trafic comme étant un élément négatif, nous montrons que ce déséquilibre est actuellement relativement bien géré par le Coroutage, grâce à la prédominance de trafic élastique (les flots TCP). De plus, ce déséquilibre dans la répartion des flots sur les liens crée une séparation naturelle entre les liens sur-utilisés avec TCP, et les autres sous-utilisés, que les flots UDP peuvent saisir et utiliser commes routes. En d'autre termes, il existe déjà une séparation de facto

entre liens sur- et sous-utilisés, et on n'a qu'à faire correspondre ceci respectivement avec la séparation TCP/UDP. Ainsi, le Coroutage tente d'être le moins complexe possible pour faciliter son implémentation à partir des réseaux et protocoles de routages actuels. Il utilise autant que possible les caractéristiques intrinsèques des protocoles de communication afin d'avoir une solution simple qui néanmoins offre une amélioration de performance.

Il reste plusieurs questions qui doivent être examinées plus en profondeur, la principale étant comment enregistrer les routes explicites. En plus des deux approches que nous avons exposé, la technologie MPLS-TE est très populaire actuellement. Cependant la question de l'extensibilité de cette technologie et de sa viabilité dans de grands réseaux reste ouverte. Finalement, les questions relatives au seuil du pourcentage ULBP, ainsi qu'à celle de la gestion des pannes, restent à approfondir. Le premier point est une direction intéressante de recherche, parce que nous n'avons utilisé qu'une valeur seuil à 50%. Les résultats avec 50% se sont avérés très bons, mais il reste intéressant de faire une analyse avec d'autres valeurs de seuil. Finalement, le deuxième point se réfère à comment le Coroutage doit réagir face à des pannes, comment gérer les routages explicites, les priorités de paquets, etc... Cet aspect est aussi une voie intéressante de recherche.

RÉFÉRENCES

[Awduche01] Awduche (D.). – RFC 3209 : RSVP-TE: Extensions to RSVP for LSP tunnels. – IETF, 2001.

[Awduche02] Awduche (D.). – RFC 3272 : Overview and principles of Internet traffic engineering. – IETF, mai 2002.

[BA et al.01] Ben-Ameur (W.), Michel (N.), Liau (B.) et Gourdin (E.). – Routing strategies for IP networks. *Telektronikk*, vol. 97, 2001, pp. 145–58.

[Bagula04] Bagula (A.B.). – Online Traffic Engineering: A Hybrid IGP+MPLS Routing Approach. *Proc. of the 5th Quality of Future Internet Services Conf., QoFIS*, 2004, pp. 2004–134.

[Feldmann et al.00] Feldmann (A.), Greenberg (A.), Lund (C.), Reingold (N.) et Rexford (J.). – Netscope: traffic engineering for IP networks. *IEEE Network*, vol. 14, n° 2, 2000, pp. 11–19.

[Floyd et al.99] Floyd (S.) et Fall (K.). – Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, vol. 7, n° 4, 1999, pp. 458–472.

[Fortz et al.00] Fortz (B.) et Thorup (M.). – Internet traffic engineering by optimizing OSPF weights. *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, vol. 2, 2000, pp. 519–28.

[Iyer et al.03] Iyer (Sundar), Bhattacharyya (Supratik), Taft (N.) et Diot (C.). – An approach to alleviate link overload as observed on an IP backbone. *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 1, 2003, pp. 406–16.

[Kc et al.03] Kun-chan (L.) et Heidemann (J.). – *On the correlation of Internet flow characteristics.* – Rapport technique, USC/Information Sciences Institute, 2003.

[McCanne et al.] McCanne (S.) et Floyd (S.). – The Network Simulator - ns-2. – `http://www.isi.edu/nsnam/ns`.

[Moy98] Moy (J.). – RFC 2328 : OSPF version 2. – IETF, 1998.

[Pham et al.03] Pham (Huan) et Lavery (Bill). – Hybrid routing for scalable IP/MPLS traffic engineering. *IEEE International Conference on Communications*, vol. 1, 2003, pp. 332–337.

[Proust04] Proust (Christophe). – An approach for routing at flow level. Internet draft (expiré). – `http://perso.wanadoo.fr/proustc/draft-proust-flow-routing-00.txt`, 2004.

[Riedl03] Riedl (A.). – Optimized routing adaptation in IP networks utilizing OSPF and MPLS. *2003 IEEE International Conference on Communications (Cat. No.03CH37441)*, vol. 3, 2003, pp. 1754–8.

[Rosen01] Rosen (E.). – RFC 3031 : Multiprotocol Label Switching Architecture. – IETF, 2001.

[Stoica00] Stoica (I.). – *Stateless Core: A scalable approach for quality of service in the Internet.* – Thèse de PhD, CMU-CS-00-176, 2000.

[Willinger et al.03] Willinger (W.) et Doyle (J.). – Robustness and the Internet: Design and Evolution. *Robust design: A Repertoire of Biological, Ecological, and Engineering Case Studies*, 2003.

[Wischik et al.05] Wischik (Damon) et McKeown (Nick). – Part I: Buffer sizes for core routers. *Computer Communication Review*, vol. 35, n° 3, 2005, pp. 75–78.

[Yilmaz et al.01] Yilmaz (S.) et Matta (I.). – On class-based isolation of UDP, short-lived and long-lived TCP flows. *MASCOTS 2001, Proceedings Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, 2001, pp. 415–22.

[Yilmaz et al.02] Yilmaz (Selma) et Matta (Ibrahim). – On the scalability-performance tradeoffs in MPLS and IP routing. *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 4868, 2002, pp. 69–77.

ANNEXE I

LE LOGICIEL DE SIMULATION DE RÉSEAU NS2

I.1 Introduction

Le logiciel de simulation de réseau NS2 [McCanne et al.] est celui retenu pour tester la performance du Coroutage. C'est un logiciel disponible gratuitement sur Internet, et de type *open source*: le code source est disponible au grand public et peut être modifié à volonté.

NS2 est écrit avec les langages de programmation C++ et TCL (*Tool Command Language*). La partie interface utilisateur est conçue avec TCL, alors que les codes relatifs aux fonctionnalités importantes et complexes sont écrits en C++. L'idée est d'avoir, grâce à C++, de la rapidité en terme de temps d'exécution, tout en offrant une interface TCL qui serait plus simple pour l'utilisateur, car à un niveau d'abstraction plus élevé que C++, pour faciliter la création et la modification de scripts de simulation.

L'utilisateur écrit un script en TCL qui définit la topologie du réseau: les nœuds, les liens, la capacité et les coûts des liens, les files d'attentes, ainsi que le trafic qui va traverser le réseau: les types de flots, leur bande passante, leur durée. Ensuite, l'utilisateur spécifie les paramètres de simulation: type de couche liaison, protocoles de routage, durée de la simulation, et enfin lance le simulateur NS2 avec son script comme entrée. Finalement, à la fin de la simulation, des fichiers de traces sont générés en sortie, qui permettent à l'utilisateur de faire une analyse des résultats.

NS2 est structuré de façon modulaire. Chaque technologie ou protocole est encap-

sulé dans un module, et l'utilisateur peut indiquer les modules qu'il désire utiliser dans son script selon son besoin. Par exemple, si l'utilisateur veut simuler un réseau sans-fil, il doit faire appel dans son script au module relatif à l'implémentation du protocole 802.11. Il en est de même pour les types de flots de données: NS2 permet entre autres de simuler les protocoles de transport UDP et TCP, les applications CBR (*Constant Bit rate*), FTP (*File Transfer Protocol*), Web, les flots à distributions aléatoires, etc...

NS2 comporte plusieurs types de protocoles de routage utilisés pour notre recherche, en particulier, les modules relatifs au routage explicite et au routage avec CCM. Le routage explicite permet d'attacher à chaque paquet de données la séquence de nœuds qui forme le chemin à suivre. Quant au routage CCM, il s'agit en fait d'une implémentation simplifiée du protocole de routage à état de lien OSPF. Le Coroutage est une méthode hybride dans laquelle les flots TCP sont routés par la méthode classique des CCM, et les flots UDP sont routés explicitement. Les paquets TCP sont donc pris en charge par le module OSPF (ou module EL), et les paquets UDP par le module de routage explicite. D'un autre côté, le terme routage standard sera utilisé quand seul le module OSPF sera activé, ce qui revient à faire du routage classique CCM.

Nous avons apporté plusieurs modifications aux modules explicites et EL, pour pouvoir simuler le Coroutage. En effet, la partie routage explicite du Coroutage requiert d'abord que les routeurs aient la capacité de mesurer le taux d'utilisation de leurs liens et de distribuer régulièrement ces valeurs aux autres nœuds. Ensuite, à l'arrivée d'un nouveau flot UDP, les nœuds concernés calculent le chemin explicite en deux étapes. D'abord, les liens avec un taux UL supérieur à 50% sont marqués comme étant sur-utilisés. Ensuite, les routeurs calculent le chemin à coût minimal ne contenant aucun lien sur-utilisé. Une fois ce chemin obtenu, la séquence de nœuds qui le compose est ajoutée à chaque paquet du flot UDP en question.

I.2 Exemple de script TCL

I.2.1 Structure de base

Nous allons illustrer la structure générale de nos scripts de simulation avec l'extrait suivant. Les lignes débutant par un # indiquent des commentaires.

```
#NO_OF_NODES = 30
#NBR_CBR_FLOWS = 4000
#NBR_TCP_FLOWS = 7000

$ns rtproto LS

$ns src_rting 1

set Turn_on_corouting 1

###Set the sizes of the link queues
set QUEUESIZE 10000
###Trace file
set tr stdout
$ns trace-all $tr
```

Les trois premiers commentaires indiquent dans ce cas qu'il s'agit d'une simulation avec topologie de 30 nœuds, 4000 flots UDP et 7000 flots TCP.

La ligne suivante est une instruction qui fait appel au module EL pour qu'il soit utilisé durant la simulation.

L'instruction suivante fait appel au module de routage explicite, pour qu'il puisse être utilisé par les flots UDP.

La troisième instruction est une variable binaire. Lorsqu'on veut simuler le trafic du script avec le Coroutage, nous mettons la variable égale à 1. Pour simuler avec le routage standard, pour comparer avec le Coroutage, il suffit de mettre la variable égale à 0.

La quatrième instruction définit la taille des files d'attente dans les liens de la topologie. D'après [Wischik et al.05], les files d'attente dans les réseaux dorsaux ont une taille très grande, ce qui nous a poussé à choisir une taille de 10000 paquets.

Enfin, les deux dernières instructions permettent de créer des traces relatives au comportement des paquets, et de les envoyer vers une source extérieure, qui est un fichier exécutable que nous avons écrit en C++, et qui permet de les analyser.

I.2.2 Les liens de la topologie

L'extrait ci-dessous définit un lien de communication. Dans cet exemple, le lien est bidirectionnel (*duplex*), reliant les nœuds 0 et 18. Il possède une capacité de 64MB/s, un délai de propagation de 50ms, et une file d'attente de type *DropTail*. Finalement, le coût du lien est spécifié comme étant égal à 1.

```
$ns duplex-link $n0 $n18 64MB 50ms DropTail
$ns cost $n0 $n18 1
$ns cost $n18 $n0 1
$ns queue-limit $n0 $n18 $QUEUESIZE
$ns queue-limit $n18 $n0 $QUEUESIZE
```

I.2.3 Création de flot UDP

Le prochain extrait de script montre la création d'un flot UDP CBR.

```
###CREATE UDP1 and attach it
set udp1 [new Agent/UDP]
$udp1 set fid_ 1
$ns attach-agent $n15 $udp1
$ns connect $udp1 $null26
###CREATE a CBR traffic source and attach it
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 120
$cbr1 set interval_ 0.015
$cbr1 attach-agent $udp1
###Explicitely route it?
if {$Turn_on_corouting == 1} {
$udp1 target [$n15 set src_agent_]
}
```

On crée l'agent `udp1` qui simule le protocole de transport UDP. L'identificateur de flot `fid` est utilisé avec la valeur 1. Ensuite, l'agent UDP est attaché au nœud 15, et est connecté à l'objet `null26` indiquant la destination comme étant le nœud 26.

La partie suivante détermine les paramètres du flot UDP. Il est défini à taux constant CBR (*constant bit rate*), avec une taille de paquets égale à 120 octets, et un intervalle de temps entre paquets égale à 150msec. Le taux du paquet est donc à $120 \times 8/0.015 = 64000bps$

Finalement, on retrouve la variable *Turn_on_corouting*. Si l'utilisateur, avant de débuter la simulation du script, a donné la valeur 0 à cette variable, cela veut dire que le trafic du script est simulé avec le routage standard. Si la valeur a été spécifiée à 1, c'est le Coroutage qui est utilisé.

I.2.4 Création de flot TCP

Afin de simuler un trafic aussi proche que possible du trafic Internet, la moitié des flots TCP transportent des applications de type FTP, et l'autre moitié transporte des applications de type Pareto. Ces flots sont déclarés comme suit.

```
#FTP FLOW
#Create a TCP agent and attach to it an FTP source
set tcp3206 [new Agent/TCP/Newreno]
set ftp3206 [$tcp3206 attach-source FTP]

###PARETO FLOW
#Create a TCP agent and attach to it a Pareto source
set tcp4710 [new Agent/TCP/Newreno]
###Parameters
set pareto4710 [new Application/Traffic/Pareto]
$pareto4710 set PacketSize_ 500
$pareto4710 set burst_time_ 500ms
$pareto4710 set idle_time_ 100ms
$pareto4710 set rate_ 100k
$pareto4710 set shape_ 1.5
###Attach to tcp agent
$pareto4710 attach-agent $tcp4710
```

Comme pour les flots UDP, une application est créee (FTP ou Pareto) et est attachée à l'agent TCP. Ensuite, ces flots sont attachés à des nœuds orgine et destination.

I.2.5 Début et fin des flots

Finalement, pour tout les flots UDP et TCP, il faut spécifier un temps de début et de fin d'émission de paquets. Pour éviter la synchronisation des flots TCP, ces derniers commencent à émettre des paquets entre le début et la $15^{ième}$ seconde de simulation. Le temps exact pour chaque flot est généré aléatoirement à l'intérieur de l'intervalle des 15 secondes. Les flots UDP, quant à eux, démarrent aléatoirement entre les secondes 15 et 35 de la simulation. Tous les flots arrêtent d'émettre des paquets à la $95^{ième}$ seconde, et la simulation elle-même se termine à la seconde 120, pour permettre à tous les paquets transitant d'arriver à destination. L'extrait ci-dessous montre un exemple de début et de fin de flots.

```
$ns at 6.128 "$ftp2458 start"
$ns at 11.466 "$pareto5102 start"
$ns at 26.615 "$cbr308 start"
$ns at 95 "$ftp2721 stop"
$ns at 95 "$pareto4734 stop"
$ns at 95 "$cbr3995 stop"
$ns at 120 "finish"
```